The Rhetoric of Mathematical Programming

Jennifer Warfel Juszkiewicz, Indiana University
Joe Warfel, Northwestern University

Transcript

Jennifer: Hey, Joe! I've been reading Jim Brown's book *Ethical Programs.* You might find it interesting. He claims rhetoric scholars need to have a deeper knowledge of software. Specifically, he advises applying rhetorical theory to understanding our network society and reconsidering our rhetorical theory based on what we learn about software (150-152). What he doesn't mention is working *with* computational scientists like you.

Joe: Well, I'm not a software developer – I'm a mathematician - so I work on algorithms rather than directly on software.

Jennifer: Do you have time to work through a concept from your field with me so I can better understand how mathematical knowledge is made? Maybe linear programming? You've told me that it's not programming of the sort Brown talks about, but that software depends on it, so that might work.

Joe: Well, it's not a type of computer programming. Although we usually use computer programs to work with linear programs—confusing, right? Linear programming is an activity; the things that you create when you engage in that activity are called "linear programs." But it's difficult to talk about linear programs abstractly. Let's actually make one, so we can talk about the components of a linear program and how I express ideas through those parts. What would you like to model?

Jennifer: You mean, we just make up a thing, and create a model about it?

Joe: Yes, it can be any system, as long as we both agree on how it works. This is just what we would call a "toy" problem.

Jennifer: How about a pasta factory?

Joe: Why not? I'll send you a pdf in a few minutes.

Jennifer: Okay, so this is a "linear program?"

Joe: Yes. This is not a computer program. It is a set of mathematical statements. When you've talked with me about digital rhetoric before, much of it seems to be about computer programs. This is not that. A computer program is a set of instructions for a computer to follow to complete a task. And this linear program contains no instructions. It is a series of statements that create a puzzle, and it provides no suggestions about how to solve that puzzle.

The purpose of this linear program is to decide how much of each type of pasta to produce. That information is encoded in what we call "decision variables." X is the amount of lasagna, and Y is the amount of macaroni.

Jennifer: So, in this first line, since lasagna is X, you're multiplying the amount of lasagna by 6, and the amount of macaroni by 10, and then adding those two products together.

Joe: Yes. That statement is called the "objective" of the problem. I'm using that word in two senses: what the pasta factory manager is trying to achieve, and also how that desire is expressed mathematically. For a factory, the objective is to maximize profit, so each one of the products represents the profit you get from each type of pasta.

Jennifer: So the profit is $6 for lasagna, and $10 for macaroni. This must be exceptional pasta.

Joe: It certainly is. The statements below the objective are the "constraints." They are the rules that you have to follow in order to run your pasta factory. To keep this model simple, we have only four constraints.

Let me explain the first constraint, which is about supply. Lasagna and macaroni use the same ingredients, and the factory has enough storage space for the ingredients for 30 batches of pasta. Then the constraint about supply is X plus Y is less than or equal to thirty.
The second constraint is about labor. The last two state that you can't make a negative amount of either type of pasta.

Jennifer: Is there anything else we should add?

Joe : That is a difficult question. In practice, choosing the objective and constraints for an application – what we call "modeling"—is often just as difficult as the math of solving the mathematical program. We need to make the model complex enough so it really represents the underlying system but simple enough so that we can analyze it mathematically and explain the results.

There is an often-repeated quote from the statistician George E. P. Box: "Essentially, all models are wrong, but some are useful."

Jennifer: The fact that models are often incorrect is a sign to me that there's something going on in terms of how mathematicians make knowledge. What kind of choices do you make to get a model to work?

Joe: Well, even in this toy problem, we made choices to include and not include certain factors. It has a lot of assumptions. For example: the ingredients don't spoil, it's impossible to work overtime, and we have infinite demand for the pasta. In a "real" problem, we would also have to make choices about which parts of the system to include or ignore or simplify for the model.

Jennifer:
Any time you make choices and establish constraints, that's a rhetorical move: by writing a program that establishes these constraints—ones that might be invisible to later users—mathematicians have built assumptions into even the simplest linear programs (Brown 150).

Joe: That certainly happens, and it can be a huge problem. When solving a linear program is included into a larger process, and something about the system you're modeling changes, it can be inconvenient or nearly impossible to go back and change the constraints. In that case, the person modifying the process may just add some constraints. The result is a linear program that doesn't represent the system that it is supposed to model because it includes rules that no longer apply.

Jennifer: Kenneth Burke describes a process like that—he uses the term "accretion" to describe this layering (*Rhetoric of Motives* 161).

Joe: Accretion is a good term for that phenomenon. It is an insidious problem that mathematicians are definitely aware of, but sometimes we don't have the time or documentation to do anything about it. Let me show you another representation of the toy problem so you can see how these constraints work, and then you may better understand why a complex system would be difficult to change.

Right now, I'm sending you the R code for a graphical representation of the linear program. Each of the lines in the graph corresponds to one of the constraints of our toy problem.

We are interested in the space between these lines. We call that space the "feasible region" or "solution space." It contains all of the "solutions" to the linear program. On the graph, it's shaded in. When we "solve" a linear program, it means that we explore the feasible region until we find one of the "best" solutions—our technical term is an "optimal solution." For this linear program, the unique optimal solution is to produce 12 kilograms of lasagna and 18 kilograms of macaroni—which is plotted on the graph.

Jennifer: Looking at this toy problem makes me realize that "real" problems must be incredibly complex since they include far more constraints, but they can have more than one solution. And then these become part of computer programs, parts of software, and it becomes difficult to tease the constraints back out again if you want to make changes or have new information about particular constraints.

Joe: Yes—although having multiple solutions isn't an issue because all the optimal solutions are equally good, so it doesn't matter which one you use.

Jennifer: Okay, that makes sense. That said, this is still complicated and important. After all, even though a linear program isn't a "computer program," it is often used by computers. And our lives are so tied up with computers. Digital rhetoricians, including Ian Bogost and Jim Brown, have been calling for my discipline to pay more attention to how knowledge making

happens *with* computers, not just through them. While we've been talking, we've been using computers and you've been sending me documents made with LaTeX and R. How do you use computers to actually help solve these linear programs, though, not just communicate about them to other people?

Joe: To actually solve the linear program, I rely on a computer program called a "solver" through use of a "modeling language."

I'll show you what I mean by using the toy problem again. Here, let me send you another pdf, made in LaTeX.

At the top, you see the linear program. Beneath it, the same linear program is written in a modelling language called AMPL.

My role in this process is to use algebra and my knowledge of the system being modeled to find the best way to solve the underlying problem with the solvers available to me.

Jennifer: So, from the simple toy problem you've created, I can see how much of digital rhetoric studies happens a level up from the kind of math you're using. And mathematicians make rhetorical choices when they determine constraints and how these constraints can accrete over time. I want to think more about this because I suspect that this can help me understand rhetorical invention differently. In particular, I'd like to reflect on the idea in my field that math provides certitude of some kind, even though what you've said implies there's far more contingency. I'll tell you what—let me send you a short message of some kind, maybe a letter, since just explaining it verbally may not be as clear.

Joe: I have thoughts on that issue as well, specifically about how the practices of mathematics contribute to that belief. I will also send you a message, but not now. This is enough for tonight.

Jennifer: Agreed. Good night, Joe.

Joe: Good night, Jennifer.

[LETTERS]

Jennifer's Letter

Dear Mathematicians (and especially Joe),

I write because I'd like to request for a further conversation. Rhetoricians, specifically digital rhetoricians, have been interested in your work with algorithms, programming languages, and how you make knowledge with math and how this math then affects the world. This is because rhetoric is the study of how we make meaning in an effort to communicate with others. Math becomes rhetorical when it affects an audience. Mathematicians then become rhetors when they make choices that will adjust the audience's response. For example, Annette Vee, a digital rhetorician, has pointed out that those writing computer code choose between writing within or around legal

strictures. Vee also brought Jeannette M. Wing's definition of computational thinking to the table, a definition that could further inform rhetoricians' understanding of how mathematicians create knowledge. Wing writes that "Computational thinking is using abstraction and decomposition when attacking a large complex task …It is separation of concerns. It is choosing an appropriate representation" (33). Our discussion aimed to further show how the choices mathematicians make are sometimes the result of mathematical strategy, but they are often the result of convention and convenience as well.

We have also explored how computers are more than tools for those in your field. Computers are able to complete calculations that are too laborious for a single person or even a team of people. This is becoming a factor of increasing interest for rhetoricians as well: N. Katherine Hayles and Richard Lanham have considered how technology has changed how we read and view text. That makes us all the more curious about how you work with solvers. From our discussion, I posit that they co-invent with you; they make possible some of the work that you do.

So the rhetorical aspects of your work are not only in the choices you make but also in your inventional process. The next step, the one on which many digital rhetoricians focus, is how that information is then communicated to an audience of non-specialists via data visualizations, statistics, or computer programs. Let's take this step together and become cross-disciplinary rather than just interdisciplinary. Let's discuss how you envision the audience of your work. To what degree does your consideration of them affect how you formulate your processes? If we address such questions together, we may be able to examine more fully the way that mathematical knowledge is made, to show that it is not a remote, impersonal method. Rather, there are scholars such as yourselves at work in a complex situation of tradition, processes, and collaboration.

So, let's talk more. Maybe we'll go out for some Italian?

With thanks,
Jennifer

Joe's Letter
Dear Digital Rhetoricians (in particular, Jennifer),
Since we talked, I've been thinking much more about the rhetoricity of my work.

The common perception of math is as a source of absolute facts; as a domain free of human bias; as, perhaps, the unique human activity in which statements can actually be proven. The base angles of an isosceles triangle are congruent; the rationals are dense in the reals; if a linear program is unbounded, then its dual is infeasible. Mathematics contains many such unequivocally true statements that can be logically demonstrated, step by step, from a small set of carefully defined axioms.

However, when we use mathematical objects to model reality, our mathematical work gains an audience—the users of the solutions of the model—and it becomes rhetorical. As rhetors, then, we have to make choices about how to describe systems that are a result of our own objectives and biases. Given a linear program, I can obtain an optimal solution, but optimization in this sense is a purely mathematical activity. I cannot "optimize" a true, living system in the world, because I cannot formulate a model that contains the world (for, if so, the model would contain itself). We talked about this—when we were modeling the factory, we had to make choices about what to include.

I've been thinking more about what we include in models, and why—and, above all, about why we mathematicians do so much linear programming. Often, we don't even consider other techniques; we simply discuss which variants of linear programming to try. But other techniques do exist. In particular, heuristic methods allow you to describe any type of relationship, not just those that can be expressed linearly; to model huge systems, including those with stochasticity; and to influence or control the structure of the solution in a way that is difficult or impossible in linear programming. However, they do not provide an "optimal" solution in the sense that linear programming does. We mathematicians have a deep desire to present "optimal" solutions to the user, even though we must admit that their "optimality" ceases as soon as they are applied outside the limits of the linear program from which they were obtained.

I would like to know more about how to understand the conversations that take place in my community as rhetorical acts; I want to be more aware of the rhetorical choices I am making, and better able to uncover the reasons for them; I hope to demonstrate to my colleagues that their choices are rhetorical, and to understand with them what the implications of that statement might be. As such, I hope that we continue this conversation.

How was the lasagna?

-Joe