SDS PODCAST EPISODE 813: SOLVING BUSINESS PROBLEMS OPTIMALLY WITH DATA, WITH JERRY YURCHISIN

Show Notes: http://www.superdatascience.com/813

Jon Krohn: 00:00:00 This is episode number 813 with Jerry Yurchisin, data science strategist at Gurobi.

00:00:10 Welcome to the Super Data Science Podcast, the most listened to podcast in the data science industry. Each week we bring you inspiring people and ideas to help you build a successful career in data science. I'm your host, Jon Krohn. Thanks for joining me today. And now let's make the complex simple.

00:00:29 Welcome back to the Super Data Science Podcast. Today on the show we've got Jerry Yurchisin, an extraordinarily clear communicator of complex topics and a world leading expert on real world applications of mathematical optimization. Jerry works as a data science strategist at Gurobi Optimization, a leading decision intelligence company that provides mathematical optimization solutions to the likes of Uber, Air France and the National Football League, Indeed, a wild 8 out of 10 Fortune 10 companies use Gurobi. He previously spent eight years as a mathematical consultant where he paired mathematical optimization with machine learning stats and simulation to inform decision making. He was also previously an instructor at the University of North Carolina at Chapel Hill, where he obtained his Master's in Operations Research and Statistics. He holds an additional Master's in Applied Math from Ohio University.

00:01:33 Today's episode may appeal most to hands-on practitioners like data scientists and ML engineers, but it does also have tons of content that will be of interest to anyone who'd like to leverage data to make better commercial decisions or optimize commercial processes. In this episode, Jerry details what mathematical optimization is, the kinds of real world problems where mathematical optimization is a far better approach than a machine learning or statistics approach. The history of mathematical optimization, including why it wasn't

popular until recently, and the cutting edge hardware and software innovations in mathematical optimization today. All right, you ready for this outstanding episode? Let's go.

00:02:15 Jerry, welcome back to the Super Data Science Podcast. It's awesome to have you here. You were here not that long ago. So, you were in episode number 723, which aired in October of 2023. So, less than a year ago. And we had to have you back on the show, because that episode much more so than almost... It's very hard to do one of these things where you're like, "More than any other episode." I don't know, there's hundreds of episodes, but it's certainly up there in one of the top percentiles of episodes that completely blew my mind, because we talked for over an hour about mathematical optimization and how useful of a tool it is in data science alongside approaches like statistics and machine learning. It's this completely other tool that you can be leveraging to solve specific problems that you're not solving optimally if you're trying to use one of those other approaches. So, wanted to have you back on and dig in even more detail today. Welcome back to the show. Where are you calling in from today?

Jerry Yurchisin: 00:03:26 It's great to be back. I'm calling in from good old Vienna, Virginia, which is just outside Washington D.C.

Jon Krohn: 00:03:32 Yeah, welcome back. Something that we talked about during your appearance last time that gave me a really crystal clear idea of mathematical optimization was this thing called the Burrito Optimization Game.

Jerry Yurchisin: 00:03:48 Indeed.

Jon Krohn: 00:03:50 And so, anybody listening right now, they can go to burrito.gurobi.com and we'll have a link to that in the show notes. And it allows you to play in this fictitious world where you're setting up burrito carts around

commercial areas, different parts of a town, and you're trying to place the burrito trucks in the optimal locations to maximize your profit.

Jerry Yurchisin: 00:04:16 Exactly.

Jon Krohn: 00:04:18 So, that scenario illustrates for me... If anybody's wondering when do I need mathematical optimization, you can just head there. It's a free thing you can try out. You just create a login and then you can play around with it as much as you like. And it provides a really clear sense of when mathematical optimization is the ideal technique for a data science problem, because there are so many constraints in the problem. So, you can have any number of trucks, you can place them in lots of different locations.

00:04:57 There are different weather scenarios. There are these external factors. All of these things that can be modeled, but are a lot... But, provide a lot of specific constraints where you might not want to fit just any number, like you would with say a regression model, but where there's some range of specific values that are reasonable. Because, you can't have negative trucks, you can't have more trucks than you have in your inventory to be able to send out. So, I don't know if you want to talk about in a little bit more detail maybe that... The burrito game, or some other kind of example, that recaps for our listeners, some of the content that we covered in that preceding episode number 723 before we get onto some new material.

Jerry Yurchisin: 00:05:43 So yeah, the Burrito Optimization Game is something that we really feel is a great way just to dive in to understanding optimization. And two of the things that I think really are illustrated there is, one, is this addition of constraints to decision making. And that's again, the main difference between mathematical optimization and

what it does versus much of what machine learning does is the predictive nature versus the prescriptive nature. Mathematical authorization is prescriptive. It falls in this decision intelligence umbrella. At least that's where we feel it fits. So, it's more about decision making as opposed to understanding the future. So, when you have these business role constraints things, those are things that restrict your decisions more or less than anything else. And in addition to that, the combinatorial complexity of the decisions, and that's something that is really highlighted in the Burrito Optimization Game where you have a fixed number of places where you can place a burrito truck to feed people lunch and your decision as a decision maker is like a, "Yes, No, do I place a truck here?"

00:07:03 So, that's what we call a binary decision variable. I can talk a little bit more about why those are super useful a little bit later. But, one thing that I like to think about is let's take a decision space like that. That's what we call the possibilities of what you can decide is a decision space or a feasible region. Those are some of the terminology that we use. If you have 40, "Yes, No" decisions, that turns out to be... I had the number in front of me, it turns out to be something like the number of possibilities is I think 1.1 times 10 to the 11th, just 40 "Yes, No" decisions. "Do I have all of them 'Yes?'" All of them, 'No?'" What all the possibilities in between, 10 to the 11th... Or sorry, maybe it's 10 to the 12th. And if you take-

Jon Krohn: 00:08:03 It's a huge number -

Jerry Yurchisin: 00:08:04 Yeah.

Jon Krohn: 00:08:05 ... comprehensively large number either way.

Jerry Yurchisin: 00:08:06 Exactly. And to bring that number into perspective a little bit, if you were to take the distance from earth to the sun in feet, that's still less, about half. It's about half. So, that's the 10 to the 11th. And 10 to the 12th is the number of possibilities for 40 "Yes, No" decisions. So, just that little 40 "Yes, Nos." So, if you have 40 spots on that burrito map, and your question is, "Do I want to place a truck there? Yes or no?" That's already more complex than the... The number of possibilities is greater than the distance from earth to sun in feet. So, that's pretty crazy. So, understanding that complexity, that just vastness of decisions, because if you're like, "Oh, I'll just enumerate through all of these, just plow right through it and figure out which one of these is the best," then good luck.

00:09:06 The earth will probably be long and exploded or something by the time your laptop is done running that. So, that's something that the optimization game Burrito Optimization Game really brings home is. "Holy crap, there's a lot of options here." And it's just for a very, very simple game of dragging trucks. And it's fun, it's interactive and everything like that. And then you get more complex with different scenarios where it's... Because, people walk to your truck and you see them filing out of little buildings to your trucks and everything like that. And, "Okay, what if I slightly move a truck over here?" And you see the nuance of decision-making and everything. And yeah, it just really drives home the complexity of what types of problems people try and solve. And again, I was talking about 40 binary variables, 40 "Yes, No" decisions. In practice when you go to... And what people are using out there in the world of business decision making and mathematical optimization, what people really use this for is the number of decision variables is thousands, tens of thousands, hundreds of thousands, millions, even tens of millions for some.

00:10:25 So, you're having all of that just massive, massive decision making capability very distinctly modeled, and just be able to click, "Go," more or less and use a tool like Gurobi to help plow through those decisions instead of enumerating them and saying, "Which one's the best?" Our special sauce solver that plows through those options in a super smart way, and says, " This is the decision that will give you the most profit for this problem, or the least cost, or doing things in the most fair way."

00:11:13 A bunch of different types of objectives that you can try and model here. And the Burrito Optimization Game just really talks about maximizing profit. But, there's a lot of utility that it can be maximized, or a lot of fairness, and things of that nature as well. So yeah, it's just a great tool. I'm super pumped that we're able to get this going last year. And we use it for a lot of events and it's just a great way to understand optimization.

00:11:45 The one thing that I do want to be... It's mathematical optimization, while it's great for that problem, it's great for so many other problems too. So, I don't want anyone to do optimization or play this game and think, "I'm not putting burrito trucks on streets, so I'm going to do something else." No, no, no. You can use it for a whole... I think last time I talked about all the different industries, different use cases, scheduling and supply chain stuff are like our bread and butter, the logistics. And there's a lot of chemical mixing like gas and oil companies use optimization a lot for that type of stuff. But, we're into financial tools, finance and healthcare stuff. We're all over the place. So, if you're thinking, "I'm in this industry, I want to know if optimization could be good for me," go onto our website, look at the use cases, ask ChatGPT, "Hey, is this a good thing for optimization?" And it will probably be like, "Yeah," because it is useful for a lot of decision making problems.

Jon Krohn: 00:12:59 Ready to take your knowledge in machine learning and AI to the next level? Join SuperDataScience and access an ever-growing library of over 40 courses and 200 hours of content. From beginners to advanced professionals, SuperDataScience has tailored programs just for you, including content on large language models, gradient boosting and AI. With 17 unique career paths to help you navigate the courses, you will stay focused on your goal. Whether you aim to become a machine learning engineer, a generative AI expert, or simply add data skills to your career, SuperDataScience has you covered. Start your 14-day free trial today at superdatascience.com.

00:13:38 Yeah, real world practical decision-making problems across, as you mentioned there, things like scheduling, supply chain, logistics, finance, healthcare, chemical mixing, those are kinds of common use cases, but it could be any industry. The key thing I think for situations where you want to be using mathematical optimization is that there's some outcome that you're trying to maximize or minimize. And, I guess, that is also something that's different about when you think about, you made the distinction at the outset between making predictions like predicting the future versus being prescriptive. And with something that's prescriptive like this, you are not taking necessarily a bunch of historical data and just trying to say, "Oh, if my inputs happen to be these inputs, what am I going to get?" What you're doing with this kind of prescriptive approach with this mathematical optimization that Gurobi offers is that you are saying, "How can I maximize given these constraints? How can I maximize some outcome or minimize some outcome?"

00:14:43 So, how can I maximize profits or how can I minimize delivery time? And then as you mentioned there, what you call the secret sauce, the key thing that Gurobi is offering is that, I guess, figuring out what your full optimization space is. That can be hard, but it's not the

hardest thing. The hardest thing is then being able to explore over that space. Which as you mentioned in the real world, there could be millions of possible decision points that could be binary or continuous. And so, the key, the secret sauce, is this Gurobi solver that can then, in a lot of situations, work extremely rapidly at optimally solving... Approvably optimally, if I-

Jerry Yurchisin: 00:15:30 Exactly.

Jon Krohn: 00:15:31 ... remember correctly. This isn't an approximation of what the maximum profit would be or the minimum delivery time. It's mathematically proven to be the maximum or the minimum, whatever you're looking for. And it typically happens rapidly, at least in my hands-on experiences with it. And critically it's easy to access through, say, Python. So, you've provided lots of, and will provide links in the show notes, to lots of Jupyter notebooks and tutorials that you've created to allow people to be accessing the Gurobi Optimizer through Python code.

Jerry Yurchisin: 00:16:10 Exactly. Yeah. And what you're saying about how the constraint part of it is a huge thing and makes things very, very difficult. And again, this requirement for binary decision variables for integer decision variables. If you're building cars, you can't build half of a car, so it may be very, very important that you're deciding things in integer quantities. So yeah, those are a couple of the key things that differentiate mathematical optimization from something that you would do purely calculus based. And you have like, "Oh, I have some curve and I want to find the maximum minimum of it, so why don't I just take a couple of derivatives and bada bing bada boom I'm done." Stuff like that, while those methods are incorporated in some things, but it differentiates from that because of all of those restrictions, and the constraints, and the ideas

that we... If I am a binary decision variable, something like, "Yes, No," do I want to open a warehouse in this city?

00:17:19 You can then have "If-thens," like, "If I open a warehouse in this city, then I shouldn't open another warehouse within 200 miles," because why would I do that? So, then you can build that logic into the model and then say, "Okay, any location I build a warehouse or any location that I put a burrito truck, I do not want to put another warehouse or whatever truck within a certain distance because it just doesn't make sense," or because of any other business reasons that you may know of yourself or people may be telling you, "This is the way we need to do things." Mathematical optimization is really just about modeling logic via algebra, and that's where the art of it more or less rests is being able to take those business problems that people are just verbally telling you and then you say, "Okay, I understand what you're saying." And then you translate it into some algebra and then you translate that algebra into code.

00:18:25 And Python is our, by and large, our most popular API, and it's really, really good. Not just saying that as an employee, but as someone who used it before joining and everything that it is great. And just make that whole process pretty seamless. And that's why mathematical optimization as a whole, I think, it's different from machine learning because of that. Modeling business logic is not something that machine learning does. If your data does not contain these cases that have happened, then you're blind first off. And even if it does, what the outcomes were, what the decisions were, all the other things that can influence such a regression, let's say. How do you know that things were being... It's just relying purely on past data is not the approach. Because what if the past was... Things have changed. There's just so many underlying things that have changed, particularly if you think of pre COVID, during COVID, post COVID type

of things. Events like that just destroy predictive models, because how can you predict what's going to happen during a once in a generation outbreak, when it's once in a generation? How much data do you have on that?

00:19:58 But, something like mathematical optimization, when you're describing the logic of a system, like a supply chain network, or a schedule, or how I want to invest in a portfolio of stocks or something like that, that logic stays the same independent among... You may want to say, "Okay, we're in a pandemic now, I want to be more conservative." You can then... "I want to be more conservative with how I invest," or something like that. Then you can really model that with logic and you can as opposed to relying on underlying data to make some decisions about that. And it's also not to say that these two things shouldn't work together. They definitely do. And that's my main message as [inaudible 00:20:47] Gurobi-

Jon Krohn: 00:20:46 You mean the two things being machine learning and mathematical optimization?

Jerry Yurchisin: 00:20:50 Precisely. Yeah. You should not be... I feel it should be very, very rare case in which you build an optimization model in which the numbers that you use there, that go into it, are not provided by some sort of machine learning process or some sort of intense, rigorous data analytics process. Be it machine learning, be it statistics, be it just really crunching some numbers and coming up with a mean of some sort. An average. That could be fine. But, there should be a lot of information that goes into that, but it's just... But, mathematical optimization itself does not rely on vast amount of underlying data.

Jon Krohn: 00:21:32 Very cool. Other than Python, what are other APIs? You mentioned Python is the most popular API, but what other options are there out there?

Jerry Yurchisin: 00:21:40 If you're into C, we have that. Java, .NET. I'm an avid R user. I love R. So, you can use that as well. Pretty much any way that you do your work, it'll be there as well. So, we definitely make sure that we are very accessible to anyone.

Jon Krohn: 00:22:04 Nice. We don't do as many R episodes probably as we should, because whenever we do, they're very popular. I don't know if you heard in episode number 779 back in April, we had Hadley Wickham on the show.

Jerry Yurchisin: 00:22:16 I did not see that one.

Jon Krohn: 00:22:18 It's one worth checking out for all you R lovers out there. It's all about R, because it's Hadley Wickham. It's pretty funny, in the episode, I have my own biases. I became a "Data scientist," before that was a term, using R, and have since made the migration to Python and mostly ended up using Python. And so, even in this Hadley Wickham interview, talking to a guy who has developed many of the most widely used libraries, even to him, I'm like, "So, how often do you use Python," and that kind of stuff. And he's like, "Never." That's his whole shtick, is taking functionality or capabilities that are in Python and bring them over to R, making sure that it is as performant on anything as Python. And also, so coming up in a couple of weeks, I'm anticipating episode number 817 will be with Julia Silge, who is an iconic R author, specifically on natural language processing with R.

Jerry Yurchisin: 00:23:25 Oh, awesome.

Jon Krohn: 00:23:25 So, that'll be a cool one for people to check out. Anyway, I digress. Yeah, so Python is the most popular API, but you mentioned also there C, Java, .NET, R. And a critical thing to mention here is that despite... Prior to our episode last year, Jerry, I hadn't really heard of Gurobi. It wasn't something that was in my consciousness. But,

since then I see Gurobi all over the place, and I hear people talking about Gurobi all over the place. And that is because people who work, particularly in a corporate setting, like at a Fortune 500 company, if I remember correctly, it was something like 80% of Fortune 500 companies use a Gurobi Optimizer. And so, while it isn't something that unlike R, or MATLAB, or Python, which are these programming languages or toolkits that you tend to learn about in university, as far as I'm aware Gurobi... You don't have that many data scientists coming up with Gurobi as part of their education. But, if they're working at a big corporate that needs to be solving these big complex problems, there's a really good chance you are using Gurobi at one of those companies.

Jerry Yurchisin: 00:24:43 Yeah. It is one of those secret things. And actually we just... Every so often we dive into the Fortune lists and compare our customers to that. So, we actually just have recent update stats and I will rattle them off here. So, of the top Fortune 500, 35% once you get up to 250, and notice the trend here. When you get up to 250, 43%, top 100 is 51%, top 10 is 80%.

Jon Krohn: 00:25:21 Right.

Jerry Yurchisin: 00:25:22 so-

Jon Krohn: 00:25:22 Right. That's where I got my 80% from. 80% of the Fortune 10.

Jerry Yurchisin: 00:25:26 So yeah, mathematical optimization and particularly Gurobi where we are everywhere in a sense, but it's just typically hidden in some application that you may use, and you just never know that you're using Gurobi, because we are the engine to the car, more or less. If your decision problem's the car, then we are the engine that makes it go. But, you don't really care about the engine. You hop in, you turn the key, and you go. And you go

from point A to point B, so you don't really worry about it. If you think about something like Google Maps, and it gives you, "Okay, to get from point A to point B here, you should take this road, and it's going to be, this is the quickest time or the shortest distance, or the most fuel efficient." There's optimization there, and you just don't even know that you're using it. And that's another example of different types of objectives that you can have in mathematical optimization.

00:26:31 Do I want a minimum carbon footprint? Do I want carbon put out there, or do I want minimum distance, minimum toll costs? All these sort of things. Anyways, but yeah, there's so many ways in which mathematical optimization is there. If you schedule a delivery from any sort of massive, one of the larger carriers, or anything like that, and they say, "Okay, well, your package is going to arrive at this time," there's a pretty good chance that some optimization went into that to help them figure out what's the way that they can get you your, whatever you just purchased, get it to you as quickly as possible, but also in a cost-efficient way for them.

00:27:12 So those are the types of things in which optimization is there, and it's there everywhere. And that's what part of our Gurobi's message for the upcoming... I guess our, we call it Gurobi 2.0 internally and kind of externally I guess, but just that optimization is everywhere, and we're just trying to highlight where it is and how you can use it and how you can use it... And again, we believe our solver's the best out there and thinking that if you do have such problems, then Gurobi might be the way to go. But overall, I think if people come away here thinking that mathematical optimization and not... Remove the Gurobi thing from it, just if that mathematical optimization is something that I should learn, then I feel it's a success today.

Jon Krohn: 00:28:06 Eager to learn about large language models and generative AI but don't know where to start. Check out my comprehensive two-hour training, which is available in its entirety on YouTube. Yep. That means not only is it totally free, but it's ad-free as well. It's a pure educational resource. In the training, we introduced deep learning transformer architectures and how these enable the extraordinary capabilities of state-of-the-art LLMs. And it isn't just theory; my hands-on code demos, which feature the hugging face and PyTorch Lightning Python libraries, guide you through the entire life cycle of LLM development, from training to real-world deployment. Check out my generative AI with large language models, hands-on training today on YouTube. We've got a link for you in the show notes.

00:28:48 Nice, and so if we have listeners out there who write Python code or write R code and they want to be getting started on using Gurobi or mathematical optimization on some real-world business problem that they have today, how hard is it for them to set up the problem? So we talked about how the hardest part of this is having the optimizer work efficiently. Gurobi handles that for us automatically under the covers. But the thing that is bespoke and different for every circumstance, for every business problem is figuring out how to set that up in our code. And so, how tricky is that? How often can somebody do that on their own versus needing to say, engage with a consultant that is expert at this kind of stuff?

Jerry Yurchisin: 00:29:35 To model the hardest of hard problems out there? It does take some experience with anything. It takes some understanding of how these particular sets of constraints work, and again, taking the logic that someone says or writes down and translating that into the algebraic logic and then that into code can be tricky. But there are some things that, specifically Gurobi and other solvers and other platforms, and things that do, be they're

competitors or something, there is sort of shortcuts out there where you don't have to know all of this stuff, and you don't have to understand all of it down to its most minute detail. So it is very easy to get started, and it is easy to, because our simple problems are extremely simple. But then, building on top of that, yeah, it's going to take some understanding, it's going to take a little bit of work, going to take some research and a lot of stack overflow, and things like that to really get to a production-level type of model, I'd say, at a large scale.

00:30:56 But the journey there is, and this is sort of one of the hang-ups of mathematical optimization in the past, is that you needed to have a PhD in order to make this journey from basic problem to actually doing something at scale for a business and making an impact. Now, that's not the case. You can just be really good at coding and understanding logic, and you can have an impact, and can solve problems, and you can provide solutions that are really doing something. And that's part of why I joined Gurobi to help get those resources out there. So I'm just going to talk about what we put out there a little bit, but there's a ton. I think at the end of the last episode I referenced an optimization book that would be really, really good again to dive into. But from our perspective, from things we released, I did two online training sessions that we call Optimization for Data Scientists, Opti101, Opti201.

00:32:07 So it was the bare basics of optimization. And then some more intermediate level, the Opti101 series, you can find on our YouTube page. And the Opti201 is going to be on our YouTube page, probably in the next month or so. And there's going to be an Opti. I think 202 is kind of how we're phrasing it or thinking about it internally, which is sort of more intermediate-level stuff. All of it has hands-on exercises, hands-on notebooks, me looking at a

camera just like this and talking to people for hours and hours, making mistakes like everyone does.

00:32:47 And it's just a lot of fun. It's a great way to just take a day or so to really improve some skills. And then we also have recently launched, I think in April, something that's a lot more massive. So, on Udemy, we have Optimization Through the Lens of Data Science. It's a four-part course. We teamed up with one of the best optimization minds out there, Dr. Joel Sokol from Georgia Tech, and he walks you through everything that you need to know about mathematical optimization, from the absolute bare beginnings to creating real models that, again, will have a real impact. And just makes that journey step-by-step-by-step, very incremental, nothing too crazy, all in Python, and then weaves in his experiences and everything with his consultancy stuff that he's worked on the side and stuff that he's done. And it is a wonderful way to set yourself on the journey. And again, it's Through the Lens of Data Science. So it's again saying how these two things really, really work well together, how they are super complimentary. And between those two things, I think you're set. But again, you can...

Jon Krohn: 00:34:22 Between the Opti101 course that's available on YouTube now as well as the Udemy course. That's Through The Data Science Lens, we'll be sure to include links to both of those.

Jerry Yurchisin: 00:34:31 Awesome. And I think we were saying, How can one person make this journey? Again, I was talking about incremental sort of building upon, "Okay, I know this; now I know a little bit more and a little bit more and a little bit more. Now I can actually get to something that makes a lot of sense." So that type of progress isn't just for learning optimization as a whole, but it's how you build a model. You start with a very basic premise, a very basic problem that someone talks to you about, and then

you build a model, and then you'll get a solution that makes no sense when you talk about it.

00:35:11 And that's 100% expected and fine. It'll say, "Oh, put one burrito truck here, and it'll serve everybody, and you'll make infinite profit," and you'll be like, "Whoa, whoa, whoa. That makes no sense. Oh, I forgot this type of constraint, or I modeled something slightly wrong," or something like that. Just building a model within itself is iterative, not just learning how to do optimization is iterative, so you're going to make mistakes, you're going to get weird answers. Like mathematical optimization, I call it, it's like the best cheater of all time. If you give it the smallest little opening to have infinite profit, it will find it.

Jon Krohn: 00:35:55 Right. Reward hacking to take the reinforcement learning terminology.

Jerry Yurchisin: 00:35:59 Yeah, precisely. Yeah. It will do that each and every time if it's possible. So yeah, if you find yourself stumbling a little bit, like, "This doesn't make sense or this doesn't make sense; why am I getting weird solutions or no solutions?" then that's perfectly normal. The best of the best of us still do that, and it is just part of the learning process.

Jon Krohn: 00:36:25 So, how do you know? You talked about that iterative process, you can get some answers that make no sense. Like you gave the example there, where you set up something like the burrito game, and it tells you, you can make an infinite amount of money by having one burrito truck that's serving everyone. And logically, you can look at that and say, "That doesn't make any sense." And so, when you've built a really poor model or there's some big hole that the optimizer can exploit, that kind of sounds like, "Okay, you can visually tell; you can logically tell this is a problem." How do you know that you've gotten to a

place with the way you've designed your model that this really is something that will work well in the real world? Because you can imagine, maybe there's some intermediate models on the way there, where to your eye, nothing seems awry.

Jerry Yurchisin: 00:37:17 Yeah, and that's where the process that I'm talking about, from written word or verbal to algebra to code. But that first part is really, really important, because when you say you have a constraint on budget or something like that, that is very succinctly and very much declared in the model. So it is there, and once it's in that model, then it's guaranteed to be respected. So if there is nothing that you can find in that translation, then you can feel very, very confident that what you're doing is representing your problem. So, and that is, I'm making it sound like it is kind of easy. It can be very, very difficult to make sure that that happens because some of that logic that I'm talking about is very complex, and it takes some experience, some understanding to really make sure that that happens. And there can be ways in which you think it's working and it doesn't, and you find out well after the fact that there is something that is kind of going awry, and there is no sort of silver bullet for that.

00:38:33 I think it is experience; it is really understanding the system that you're modeling and what makes sense. So the burrito game, making infinite profit, obviously that's a clear indication, but maybe if you are someone who is in that business and really understands food service and putting out food trucks and fleet management or something like that from that perspective, they may understand something that you didn't and then therefore saying, "Oh, this kind of doesn't make sense; you should really add a constraint to make this happen," or something along those lines. So I would say that SME expertise, Subject Matter Expertise on the actual problem that you're solving, is probably the best thing to really

make sure you understand that the solutions that you're getting are in line with what you would expect.

Jon Krohn: 00:39:39 You mentioned earlier on about how when you're setting these things up, in addition to the kinds of tutorials you provided, the Jupyter Notebooks there, people might want to consult things like Stack Overflow. Now that immediately in my mind jumped to today how I am much less frequently using Stack Overflow, and for me, I'm usually using Claude by Anthropic, but there's also GPT-4o from OpenAI or Gemini from Google, these state-of-the-art LLMs. Have you kind of pair programmed with one of these LLMs on an optimization problem?

Jerry Yurchisin: 00:40:17 Yes, and I do it a lot to see how well it works. And actually, that is something that I'm exploring with people internally. We are putting together a custom GPT that will help with some of this, and we're coming at it from an educational perspective of this is going to help you understand optimization modeling. I would not feel comfortable putting anything, copy pasting code, and putting it into production right now for anything, really. I mean, not just optimization, but anything, literally anything, any code generation that pops out for anything, you got to do some checking on that. Same thing for mathematical optimization, but it does a surprisingly good job of making that logical connection from... There is this incredible parallel between how I was talking about someone states a decision problem, and you need to translate it. That's exactly what you can do with all the tools that you were just talking about.

00:41:36 And so you just give it a prompt, and yeah, it'll do a really good job. There are things that it makes mistakes on. And we are trying to understand those, and I think it could be later this year. We do plan on sort of doing a webinar on that. It'll be something that'll be really cool. Again, I'm experimenting a lot with some problems, and we also

have a couple other people who are doing that. And part of Gurobi's innovation is, we had an internal innovation competition, so everyone submitted ideas and things, and one of my colleagues was like, "Hey, we should do modeling with GPT." Everyone's like, "Yep, 100%." No other innovation project really came close. That was the clear winner. But yeah, it's something that I 100% think is a great idea. You just need to take it with the proper grain of salt that you do with everything.

Jon Krohn: 00:42:43 So for getting started, for educating yourself on how optimization problems work and how you can be integrating Gurobi through the Python API or the R API into your code, having suggestions from these top-of-the line cutting-edge LLMs, like I already mentioned, Claude 3.5, GPT-4o, and Gemini from Google, I suspect it is similar to my experience with any of the coding that I do, where, yeah, like you said, most of the time it doesn't make mistakes, but because it does sometimes make mistakes, you need to be sure that you understand what's going on in that code. You can't just copy paste and put it into production like you said.

Jerry Yurchisin: 00:43:27 And from our experience, the simple models, we actually started benchmarking these too, our simple models from our Jupyter Notebook library, and it does super simple models extremely well. You can set it and forget it for that. And then once you get to intermediate, there's like a coin flip, whether or not there will be an error. And then complex models, right now, we've seen some improvement, but there's just some things that it just... Particularly once you start talking about abstract ideas, one that we had a little bit of a problem with but actually recently saw some improvement is a 3D Tic-Tac-Toe type of game. Okay, can you do this? Filling in minimal number of lines in order to do something. I think it's filling in the 3D board with minimal number of connecting three Xs in a row or three Os in a row, doing

that. And we had some problems with just the abstract nature of that problem. It just didn't really translate well. So I mean, yeah, there are some things, there are some problems, but yeah, always verify.

Jon Krohn: 00:44:47 Nice. All right, so on the topic of LLMs, companies like NVIDIA have had explosions in their share price because GPUs are critical to efficiently training any kind of deep learning model. And the larger they get, so huge LLMs like the ones I just mentioned, GPT-4 kind of class models, Claude-3 class models, Google Gemini, these are gigantic; you're going to need many GPUs to train over. And so, the most cutting-edge, like H100 NVIDIA GPUs, are in really high demand, because if you want to be training the next generation of LLM from scratch, you're going to need the absolutely most cutting-edge hardware.

00:45:36 And so these kinds of processors, GPUs, they build on the same kind of, they're called graphics processing units because they originally were for rendering 3D graphics and things like video games or when you're doing video editing on your computer and the same kind of simple matrix multiplication that is critical to doing that kind of graphics rendering also turns out to be the kind of highly parallelizable simple computation that we need for training or even at inference time with deep learning models. And as I already said, as they get really big LLMs, the more and more critical having GPUs becomes. If you tried to train an LLM on CPUs from scratch, it would... Like, you gave the example of the earth exploding or maybe our sun going nova before you would have your model trained. So a question that I have for you is: how does mathematical optimization relate to the kind of device that we're training on, like a CPU or GPU?

Jerry Yurchisin: 00:46:47 So, for the question of GPUs, hey, GPUs are the cutting edge for everything. Is that the same thing for mathematical optimization? The answer to that is no; it's

not the right tool for the right job, which was something that I harped on, I think last episode. Choosing the right tool for the right job for decision problems. Well, for this, it's like GPUs, again, Jon, as you were saying: massive parallelization, simple operations. That's just not what the algorithms that we have for mathematical optimization, those are not like that. They're not hyper-parallelizable, they're not providing simple computations by and large. So if you do try, and right now if you try and run typical mathematical optimization algorithms, which I can talk a little bit more about those in a little bit, sort of high level on what goes on there. If you try and do that, then it's just not the right fit, and you'll suffer some performance issues there. There is one little exception in matrix factorization, is a very important part of one of the types of algorithms that's used, and that's something that can be done pretty well on GPUs.

00:48:14 So there is some hope, and it is something that the team Gurobi, and other folks are trying to look into. We're always looking for the best way out there to run what we want to run, to solve the problems that we like to solve. So, if, at one point, GPUs are the way to go, then that's the way that we're going to go. So it's that being said, yeah, CPU is the way to go. But one interesting thing that I do want to mention about, particularly with NVIDIA, is they actually, through a customer, through something, through the fates that be, came and sort of knocked on our door and were like, "We heard you have really hard computational problems for CPUs." And we're like, "Yes, we do." So they wanted us to test out their Grace CPU on the problems that mathematical optimization solves and use Gurobi. So one of my close colleagues, we call him the mad scientist, Greg Glockner, who's a VP technical fellow at Gurobi, has been working at Gurobi for, I think, he was employee three or four or something like that.

00:49:43 He's number one in my book. But he did a lot of testing with other folks on our side and using their CPU compared to sort of the established AMD processors that we typically use for benchmarking. We found a 23% improvement on hard problems while using 46% less energy. So that type of improvement, like, "Hey, that's a pretty impressive improvement from a speed perspective, but from an energy consumption perspective," it was something that was really interesting to see. So the CPU environment for us, we're still improving; there's still cool things happening out there. And from, I guess, a computational perspective of, "All right, how much faster is this stuff getting over time? Are you guys reaching a plateau? Is the hardware reaching a plateau? Are the algorithms sort of slowing down?" The answer to both of those is not really, and as I was just talking about, there is still room for CPU performance improvements there.

00:51:09 There's room for energy improvements, energy consumption improvements, and what we work on is the algorithmic improvements as well. And I actually just ran across this slide that we have for presentations, just earlier today, from version 11, which was released last November. To comparing that to our earliest versions, where the Gurobi solver, independent of hardware, is 80 times faster. So if you think about that, 80 times faster than we were 10 years ago, plus all of the crazy hardware improvements, processing improvements on top of that, I mean, you're solving problems like thousands times faster than you used to be. So stuff that would take, "Oh, this would take a day to solve or to take a week," is solving in minutes and seconds now.

00:52:06 So that's part of what we want to get out there into the world as part of our message is, "Hey, CPUs, processors are awesome and they're getting better." We got really smart people and we bring more on. Year after year, we're expanding our teams and bringing in the best of the best.

Our algorithms are getting better. Because of that, problems that you would just not even think about solving five years ago are solvable in minutes now. And that is a key thing that was a common misconception of mathematical optimization, is, well, it's too complex of a problem to solve. The computational stuff, don't even worry about it. And we're like, "Hey, at some point, yes, you can make problems infinitely large, and yeah, it's not going to solve, but you can still get real business results, solve real problems at the scale that you want and it's doable." So that's part of my spiel, part of my getting on my soapbox, is that.

Jon Krohn: 00:53:13 Well, what's really interesting about this whole conversation that you just had is that I was only vaguely aware that NVIDIA was working on CPUs at all. Now that you say that, it's something that there's some cobwebby memories out there, but this is the first time. So the blog post that you provided authored by Greg Glockner, whom you mentioned earlier from Gurobi, that VP and Technical fellow at Gurobi, I will be including in the show notes a link to this blog post, which is super interesting because it's going into so much detail on NVIDIA's CPU's. And it's got it right in here, as you mentioned, compared to the AMD chip that I guess you would typically be using, it's 23% faster and uses 46% less energy. So effectively half the amount of energy, which when you think about the scale that these things can be implemented on, even just work that Gurobi is doing where it's greater than 1,000 clients, something like 1,200 clients that you guys have around the world. And so that cost savings is gigantic, and energy savings is gigantic.

Jerry Yurchisin: 00:54:22 Yeah, exactly. And as a company who preaches like, "Hey, you want to reduce costs, use the mathematical optimization." If you want to reduce costs, then there are things out there that can help run our product and other similar things faster and more efficiently.

Jon Krohn: 00:54:43 And just reading some more details about this Grace CPU from NVIDIA, it has 72 cores and 480 gigs of high performance low power memory. So that's wild. I mean, it's taking the same philosophy I suppose that NVIDIA has had historically with parallelization, lots of memory, high speed memory transfer. So that expertise that they have around developing the world's by far most popular GPU's, AI inference accelerators, taking that same expertise and now applying it to CPUs, making chips that have tons of cores and tons of memory. Cool. Great story there. In addition to new hardware options, like being able to use those kinds of CPUs, you mentioned things like an ADX speed up for the Gurobi software. Just the way that the Gurobi optimizer is implemented, you get ADX speed up hardware. Hadn't changed over the last 10 years, you'd still enjoy that ADX speed up because of more clever work on the software side of the optimizer. So can you tell us more about new optimizer versions that you have and maybe go into more detail on how the main algorithms work?

Jerry Yurchisin: 00:56:02 Yeah. So what we've at Gurobi have been diving into a lot over the last couple years is if you look at what MIP stands for at MIP, that's Mixed Integer Programming, but there's an implicit L there, and the L is for Linear, Mixed Integer Linear Programming. So everything needs to be linear or it should be linear. It's very helpful if it's linear. So your constraints need to be linear functions. Your objective needs to be a linear function for that.

Jon Krohn: 00:56:34 This reminds me, I think a year ago in that episode, in episode number 723 that you were on, this reminds me of MILP, Models I'd Like to Program.

Jerry Yurchisin: 00:56:45 Yeah. I remember saying I need to use that and I think I used it for a little bit and then I forget, but I'll bring it back for sure. So a lot of this stuff is really helpful to be everything linear fashion, and the complexity would be in

the integer variables, the binary, stuff like that. That's where things got complex and things. Now we're diving into quadratic things. So your objective is a quadratic function, your constraints are quadratic. And that was the big thing from I think a couple versions ago for us, was it was really doing well in that. And those types of applications are very important in chemical engineering and things like that.

Jon Krohn: 00:57:38 And so in case people aren't aware off the bat, if you can't quickly visualize what that means, quadratic, it means that as opposed to a linear relationship, so with a linear relationship as X goes up, Y say always goes up, or as X goes up, Y always goes down. So you have that linear relationship between X and Y. With a quadratic relationship as X goes up, there might be a period of time where Y goes up and then it starts to go down or the other way around, it begins by going down and then starts to go up. So you have a curve that can be modeled quadratically with a squared variable. So obviously there's lots of real world scenarios where you need to be able to model that in order to build a high quality model of some real world process. So I think you were just starting to mention that. Was it with chemical applications?

Jerry Yurchisin: 00:58:31 Yeah, chemical engineering is one in which that's a go-to way. And prior to that you would do a lot of what we call linearization of things where you would estimate those non-linear functions with piecewise linear approximation. So instead of your nice parabola, you would have line, line, line. That mimics it, and that is good for some things, but also you lose precision. And the more precise you would like to make that piecewise linear representation, the more additional variables you're adding to the optimization model, which makes it run not quite as quickly. So there is a balance there, precision versus runtime. But now with our last version 11.0 and

what we're getting into in 12, which is going to be coming out in a few months, is just more general non-linear and what we call non-convex optimization. The assumption of convexity, which is if you think of a feasible region, it's convex if there's no dents in it.

00:59:42 Take any two points, draw a line between those two points. If every line that you could draw between any two points is completely in the set, then that's a convex set. And if you don't have that, then some of the previous stuff I've been talking about of mathematical optimization, it just gets really difficult, somewhat falls apart. But we understand that those problems are the ones that people want to solve, so the non-convex stuff is something that we've made a lot of strides on. And then just incorporating more non-linear functions that you can use and not piecewise approximating them. So not approximating them with a bunch of straight lines that mimic curves, but the actual curves themselves and implementing algorithms that solve those, which I'm still getting myself up to speed on what those are doing.

01:00:31 But just being able to solve more realistic problems at the speed that people need to solve them is what we're looking into and what we're trying to improve on version after version. So our bread and butter is that mixed integer linear program, and we're still always having improvements there, but we are also looking into these non-linear, non-convex problems as well, which are everywhere as well. So we are trying to become the solver for everything. And I think one approach that we are trying to take to that that's a little bit different than maybe what some people would think would be our competitors, is there are some other tools out there that do a lot of local optimization for non-linear problems, and they don't worry about things like the mixed integer portion of your decisions. We are still very much what Gurobi is about, is this global optimality.

	01:01:38	We are provably giving you the best solution for your problem. May take a little bit longer in some aspects, in some problems, but we feel that that is something that is super important. I think I gave an example last time of if you're a major airline and you could reduce your fuel costs by 1%, that's massive. If you're just any bigger company and you could reduce costs by the tiniest bit, that's really good. So having that global optimality, that's something that is guaranteed with how we approach optimization, is really important to us and that's what we want to put into our products.
Jon Krohn:	01:02:20	Nice. So handling more variables like quadratic relationships as inputs or outputs as well as being able to always model globally across the whole decision space as opposed to just locally, are key elements there for you. So tell us a bit more about why integer variables are so complex to model. So that isn't something that would necessarily be intuitive to me.
Jerry Yurchisin:	01:02:49	You would think that if I have, I guess everyone in your minds, let's do a mind exercise... have a two-dimensional graph and just put some sort of polygon on there and make it convex. I just gave you the idea of what convexity is, so no dents in it. There's no dents. So just draw a polygon in your head, and then within your head highlight the integer dots. So where one and one cross, where two and two cross, where one and two cross and everything like that, you put a dot instead of just having a shaded region. The shaded region is what we would call a relaxation of an integer program. The dots are the actual points that you want as your solution. So you might be thinking, "Well, if I have less points, then it should be easier to solve." And it's exactly the opposite.
	01:03:52	Essentially, the way that the main algorithm works for solving linear programs, it's called the simplex method. If you have this polygon in your head, what it does is it

actually, because of the math behind it, the optimal solution must occur at a point in which two of the outside lines meet. So essentially, a vertex of your polygon, that's where the optimal solution must be. It can't be anywhere else. Well, I'll take that back slightly. It can't be on the inside. You can have multiple optimal solutions if you have two points and then the line connecting them are all going to be optimal with the same objective value. So you have your choice in a sense, but it's going to typically be one of those corner points of your polygon. So because of that, because it's guaranteed that that optimal solution is going to be on the outside there, all of those integer points that you have on the inside are probably not going to be right there.

01:05:03 So unless you just happen to have that be the case, now you need to take the main algorithm that searches the outside of your polygon, searches it very smartly. Now you have to do something else. And the way that that's done for the mixed integer programming is it takes that problem and it breaks it into the main algorithm that is used for that is what we call branch and bound. But there's another one I can talk about in a second. Branch and bound, essentially what that does is let's say that you have this polygon and your optimal solution is where X is 1.5. We know that we want an integer value for X, so let's set up two sub problems. One where X is less than or equal to one, and then one to where X is greater than or equal to two. So you're splitting off of that, you're branching off of that variable because we want X to be an integer and it can't be 1.5.

01:06:07 So let's say it's either going to be less than or equal to one, greater than or equal to two, and then we solve more linear programs with the simplex method. And you keep doing that until you get to a criteria that says this is your mathematically provable optimal solution. So essentially that's part of why it's more complex to have that integer

stuff is because you need to run possibly exponentially many linear programs. What I mean when I say linear program, where you don't have this integrality restriction. So you let loose the rules a bit, solve, and then you implement more rules as you go. So that's the iterative process that happens. So eventually you'll get to something that says all of my decision variables I want to be integer are integer. I have a couple other things that happen about lower bounds and upper bounds are meeting. And then because of all of that, boom, it's math says that we're at an optimal solution.

Jon Krohn: 01:07:11 So I'm gathering here that the fundamental problem when you want to have integers in your model is that you can't do calculus jumping from point to point. Calculus only works over a curve.

Jerry Yurchisin: 01:07:22 More or less, yeah.

Jon Krohn: 01:07:24 So I think that's fundamentally the idea here is that you come up with ways of artificially constraining things so that you can work over a curve. So by creating these relaxing constraints, like you said, but doing that a whole bunch of different ways, you're able to view a problem from multiple different perspectives around a point and say this individual point actually is the best from all the possible points out there.

Jerry Yurchisin: 01:07:58 Yeah. And I said something like, "Then the math says you're at an optimal solution." So it is worth maybe diving a little bit into how that happens. So going back to a linear program, so let's say all of your variables are continuous, which makes it easier. So the polygon in your head is all just shaded and don't worry about the individual dots in between. So worrying about the corner points and stuff like that, that's a linear program where all your decision variables are continuous. For every linear program, there is something that's called a dual

problem. So it's just another representation that's like a mirror image I guess, in a sense. If you're maximizing your regular problem, the dual is a minimization problem. And the math behind it essentially says that you have your regular problem and your dual problem. If the objective function value at the optimal point for each of those problems, one going up, one coming down, is the exact same value. And that's proven with fancy math and proofs and stuff like that. And to use the terminology, the original problem is called primal, and your dual problem is called the dual. So your primal problem for its optimal solution has a particular objective value.

01:09:45 The dual has the same objective value, but different variables. I won't go into details of how they translate to one another, but it's pretty easy to actually go between one and the other. But essentially one's going up, one's going down. When they meet, that point in which they meet is your optimal solution. And so that's part of a little bit the math behind it. How we can say "we guarantee" is because people have proven with the fancy math that if this happens, then that's how you know that a linear program is giving you the optimal solution without having to exhaustively search all points and exhaustively search everything. And when I say stuff like the smart math, the smart things that are happening, that's a little bit of a glimpse into how that works.

Jon Krohn: 01:10:39 Nice. Very cool. I am always learning a ton from you, Jerry. You're a great explainer of complex concepts and you're great at creating visuals as well. It's interesting, as we're recording, I'm not usually closing my eyes and imagining polygons.

Jerry Yurchisin: 01:10:54 I thought you were taking a nap. I thought I was boring.

Jon Krohn: 01:11:00 So onto another tricky technical question for you that I'm really curious about is NP hard problems. So

definitionally tricky. So tell us what NP hard problems are for those of us who don't know what they are and why we shouldn't just ignore them. And then critically, why NP hard problems that people wouldn't have even tried five years ago can now be solved in seconds with mathematical optimization.

Jerry Yurchisin: 01:11:32 So my quick little rundown of computational complexity is you may have heard something like, "Is P equal to NP?" That type of argument. And so essentially what that's talking about is the algorithms to solve problems. What is their computational complexity? So if something is in class P, that means that there is a polynomial time algorithm that will give you the solution. And those are nice, those are easy, that they run quickly and everything's pretty good with. That's where you'd like to be. Then NP is what we call non-deterministic polynomial, and essentially there's no polynomial time algorithm known to solve it, but if you're given a solution, you can quickly verify that it's correct. So to actually solve the problem, very difficult, but to verify can be easy. Then after that is what we call NP complete, which is the most difficult of the NP problems. So if you're able to come up with a solution or an algorithm that solves one of these quickly, then you can solve all the other ones in the whole class very quickly. So it's like the domino that would make life easy for everybody in the computational space, but that's probably not going to happen. But then there's NP hard, which is the same thing. But unlike the NP complete, you don't necessarily be able to have to verify a solution quickly. So it may be actually very difficult to verify if you're asking it, "Is this solution the optimal solution?" That may be hard to just find out on its own.

01:13:29 So that's where actually mixed integer programming is, is that NP hard. So you hear about this thing, "These problems are so difficult to solve," and yes they are. But the thought is, if it's in this category, don't even try to

solve it with an exact solution. Which is something that, again, like I say, it's just what Gurobi provides and what mixed integer programming, the solvers like us, what we provide is that global exact solution. We need to use heuristics. We need to use approximations. And so it's just a scary word or a scary phrase or something and it just turns people away from it and like, "I'm not even going to try. I'm not even going to attempt." And I came across this. I was doing a webinar and I was talking about, we have essentially a Python package called Gurobi Opti Mods, which is prepackaged optimization problems where you just feed some data, it runs, you don't have to worry about any modeling and it gives you an optimal solution.

01:14:38 So it's very cookie cutter problems. And one of those is what we call a maximum weighted independent set. And I won't worry about going into that. You can watch the webinar on YouTube and find out for that yourself, but I was looking at the documentation for a Python package that claims to solve this problem, and there's a line in there that just says, "The actual problem of this is known to be NP hard," so you're just immediately better off using approximations, using whatever, using some heuristic to find the solution. It just immediately put its hands up. And this was documentation saying that, and it was documentation for the package itself. So I guess, sure, it's not going to say use other stuff, but whatever.

01:15:35 In that webinar, I just gave it a very small problem of taking nodes in a network and finding a subset I think that covers all the arcs. I'm blanking on it now, but again, just watch the webinar. But it was trying to solve this very simple version of it and it was just a 10 node problem and it was giving wrong answers. Looking at the graph, I could visually see that it was wrong and it was giving suboptimal answers. How would you think about how this would perform at any type of scale, 100s, 1,000s

of nodes. You're trying to do some social network analysis and you're running this package and it just giving you clearly suboptimal solutions. So there is this just NP hard fatigue. And then that translates into mixed integer programming of, "It's a very hard problem. Don't try solving it with a solver like Gurobi because it's just so complex, it's just never going to work."

01:16:46 But that was the case like 5, 10 years ago. And again, all the stuff that we were talking about before with our algorithmic improvements, hardware improvements, it's just the things that were that's just "not even worry about it." Oh, we have a mixed integer programming problem with 10,000 variables and people are scared of that and thinking, "That's impossible to solve. You'd never solve that in any type of time that would make sense." Those are being solved in half a second, a second, instantaneously nowadays. So it's just yes, that is true.

01:17:20 If I kept the whole thing behind this computational complexities, if you expand the set of inputs, you're growing exponentially. Yes, eventually you can grow the problem such that it'll take Gurobi forever to solve something. That is true at a certain size. But again, now we're getting to the point with all the hardware, the software and everything where real problems are now manageable, real problems are now solvable in real time for some things. Sometimes, you may have to concede a little bit of that realism to get a little bit of performance or something like that, but that's trade-offs you make for everything. You'd make that with machine learning training. I want to be able to retrain models quickly, so you'd make some sacrifices there, things of that nature. So there's always that balance for everything, but with mathematical optimization, yeah, there is that stigma and part of our message is, "Hey, try us again, then you may be pleasantly surprised." And what I will add to that is there is a difference between, I may have talked about

this last time, between a commercial solver, yes, Gurobi eventually you do have to pay and buy a license from us if you want to use what we have at the right scale and everything. Yep, that is true, but we have the best minds in optimization building that for you. So yeah, they're not doing it for free yet. I asked, they say no. But you can use open-source solvers and I think that's a great way to try and solve real problems at a smaller scale and get yourself going and try and understand, "Hey, does this have business value for me? Is this going to be helpful?"

01:19:14 Yeah, you may have to condense things, but it's a good way to learn, good way to get started. And then by the time you would need something like Gurobi, your problems probably have expanded to a point in which it is worthwhile to save that, to have something that takes us open-source solver maybe days or weeks to run. And we've had that sort of happen where a now customer or someone who's trying to evaluate us would say, "Yeah, this with an open-source solver would literally take a week to run. We just click go on it and just came back when it's done a week later it would be there." Now it's solving in 10 minutes, 20 minutes or something like that. Something that used to take a day or it would take a day with open-source, now takes 20 seconds, 10 seconds to solve with a commercial solver like us.

01:20:05 So part of that is yes, the problem itself is NP-hard. It's very difficult. So if you were to attack it with something that is an open-source solver, keep that in mind that there are options past that. But I don't want to discourage the use of open-source, because it is the perfect way, a great way to really, if you want to learn, it's a great way to use something that's free to understand the value to your business. Right now something like Gurobi you can download, if you PIP install Gurobi, you can use a 2000 by 2000 sort of trial license to get yourself understanding, again, that type of... When I say 2000 by 2000, it sort just

comes out naturally to me, because I know exactly what I'm talking. 2000 decision variables, 2000 constraints.

01:21:05 So when you think about the number of burrito trucks that you're putting out there and the number of constraints that you're adding to that, yeah, it's a fairly small problem, but it's really great way to help you learn and understand and sort of just build a small scale thing that says that that's somewhat representative of your problem. And then if you need to expand, then hit us up again and we'd be glad to give you free evaluations and help you work through that as well. So there's a couple of things there that I wanted to mention about if you try optimization and it fails not to think about why it might be failing and if you are using an open-source solver, many of them are really good. Some of them could be good for your problem and you may never need anything like Gurobi, that's certainly possible. But once you get at scale, there's a decent chance that you may need something like us.

Jon Krohn: 01:21:59 Nicely said, and that all made perfect sense to me, the open-source trade-offs versus using a commercial solution like Gurobi, particularly when you get to at scale. But the way that you got into this was we started by talking about NP-hard problems.

Jerry Yurchisin: 01:22:13 I deviated.

Jon Krohn: 01:22:16 And so just I wanted to say that, because I don't think you mentioned this, that the NP in NP-hard stands for non-deterministic polynomial time problem. And it's kind of another way of saying a very complex problem, because there isn't a deterministic for sure. You're not going to be able to follow the same path to get the exact same answer every time. Polynomial meaning things like having quadratic relationships, not just linear relationships in there. And so I don't know if you happen to have off the

top of your head, Jerry, like real world NP-hard problems that maybe a few years ago no one would've dared to try to tackle, but now you can tackle potentially in seconds with Gurobi.

Jerry Yurchisin: 01:23:02 It's a little bit difficult to say one type of problem, because all of MIPS, Mixed Integer Layer Programs, they're all kind of similar in a sense, whether using them in supply chain, whether using them in finance, whether using it scheduling, they all sort of translate to the same thing at some point. You take your decision, you take the verbal problem and you put in algebra. Once you get in that algebraic form, they're very similar in what you would see sort of written down pen and paper. So it's kind hard to distinguish that. But at the same time, operations research folks, they love to talk about sort of problem archetypes. So very common things like the knapsack problem is how much stuff can I fit in knapsack to maximize its utility before I go on a hike? Something like that.

01:23:53 Another one is the traveling salesman problem, and this is something that you see a lot of... You might see some neural networks trying to tackle this problem. You might see quantum optimization trying to tackle this problem and obviously mathematical optimization, Gurobi trying to tackle this problem. It's just a very common easy to understand problem. And traveling salesman problem if you're not familiar is you have a set of cities that you want to visit, the salesman wants to go and sell stuff to each of these cities. What is the shortest path I can cover all of those and then get back to my starting place? What's the shortest path that I can take? A common thing is I want to travel to all 50 or say all 48 state capitals, what's the shortest path that I can take? You don't want to drive from New York to California to Florida back to Washington. That's obviously not great.

01:24:52 You want to find the shortest path to travel all those things. So that may be sort of the problem that people go to as like this is an NP-hard problem and it's sort like the go-to, this one's very difficult, because it is difficult to solve. But I think now this is where I might have to have to come back and do some research and stuff like that, but you can solve problems that the key metric I guess or the key sort of quantifier of a traveling salesman problem is the number of cities. And there would be, if you want to do a 50 city problem even five years ago, 10 years ago, that's something that would be like, "Oh man, that's kind of difficult to solve." Now we're into thousands and stuff like that where you can easily solve a traveling salesman problem with that type of size relatively quickly or at least quickly enough that makes sense for whatever real application that you're trying to solve. So that's sort of the go-to thing that people like to talk about.

Jon Krohn: 01:26:13 I see. My last topic area for you, you've been very generous with your time today. We've gone well over the recording slot that we'd agreed to. But one last topic area I have for you is around the history of optimization. So it relates to the same kind of thing. You just said that a few years ago you might have stopped the number of cities that you might've tried to fit into the traveling salesman problem might have been 50 and now it's thousands. So can you stretch back a bit further with the history of optimization and tell us why it initially wasn't popular? Maybe what some of the first use cases were and how that brought us to where we are today?

Jerry Yurchisin: 01:26:52 There's sort of two big names and I have it in front of me so I don't mispronounce it. Kantorovich in 1939 and then George Dantzig in 1947, those were two of the people who really brought linear programming as I was describing or earlier brought that sort of to the forefront. But Dantzig gets a little bit more of credit. He's sort of like the name, because he was the person who invented the simplex

algorithm, what I was talking about before, going from corner point to corner point in a polygon. But using that algorithm to solve a linear program sort of brought the linear program as an important planning and decision making tool around that time. So one of the first applications of that is if you ever go to an OR, Operations Research site or something about mathematical optimization and it's like here's our first example. It more or less is going to be something called the diet problem. And that was a very small problem that was developed for I believe the US army to help.

01:28:12 How can I feed troops? How can I feed a battalion or something like that? How can I feed that group of individuals making sure that they have the nutrition that they need but at minimal cost? So that was one of the original applications of linear programming way back when. Things like and I talk about the integer programming was developed more or less in the mid '50s with Dantzig and other folks as well were adding to that. So around that time was when a lot of the initial theory was developed. The problem was once you tried to get past that real simple problem, which was nine constraints in 77 variables, so that's a very, very small problem, exceptionally small.

01:29:11 Once you get past that, the computational power just did not exist. So fast-forward to maybe something like the '70s was when we actually started to get a little bit more of the information, a little bit more of the power to actually solve some of these slightly larger problems that were still relevant to businesses, still relevant, but we actually be able to solve them in any type of speed that would make sense. It's not taking days or weeks or years, but something that you can actually use. And a lot of these problems were for oil and gas companies, for refineries was one of the original adopters of linear programming and they were actually very much into

funding this type of research in order to push forward the theory and the technology and everything like that.

01:30:10 So essentially what the main bottleneck was, was computational power, and that is something that is very, mirrors a lot of why it took so long for deep learning to also become the powerhouse that it is today is because for deep learning, the data wasn't there that we needed to exist to really use it, but the computational part wasn't there as well. Enter GPUs and all of a sudden boom, this whole new thing sort of just explodes. And that's sort of what mathematical optimization kind of hasn't had just yet.

01:30:51 I mean, obviously I've been talking a lot about how the CPU and performances and all that stuff has really increased over the last 20, 30 years and stuff like that to problems that actually that were never be able to be solved, now they can be solved in minutes and seconds. But there wasn't a new technology that came on the scene to really spur it like GPU's did for deep learning. So it sort of just got kind of lost in the shuffle a little bit. It definitely found its niche. Niche is too narrow of a word I guess, but it has its early adopters. So I mentioned supply chain a lot. That's sort of one of the big ones that became an early adopter to mathematical optimization and showed a lot of business value there. But then once you get into the '90s and things like that is where a lot of people started putting effort into the algorithms and really improving that as well.

01:32:07 So there's some groundbreaking research that really sped up that part of it and just going, adding to that time over time over time. And then Gurobi came on the scene in 2008 from a company one of our competitors called CPLEX, which is owned by IBM. They were a bunch of people there, including all three of our founders were at CPLEX and making awesome advancements in the LP

algorithm space. But then just decided like, "Hey, we're getting a little bit slow. We want to take a different approach." So that's where Gurobi came in. So our three founders decided to start a new company where creating the fastest, most powerful mathematical optimization solver that we possibly can became the top priority.

01:33:15 So yeah, there's a lot of early adopters from a theoretical perspective saying, "This could be really cool." Just the computation stuff wasn't there, it got there a little bit. Oh, people are like, "This is kind of nice, but again, let's solve more realistic problems." Again, sort of entered that same bottleneck of computational inefficiency and now that things have been steadily increasing from people making the algorithms better, from people making the hardware better, again now we're actually at a point where over the last few years really seeing businesses solve awesome problems that are at the scale that they need.

Jon Krohn: 01:33:56 Nice. Great recap there. It's awesome that you had prepared notes so you could speak so specifically about dates and people, significant events, fantastic. No doubt there will be more mathematical optimization in the future as more and more people learn about it through things like this podcast and as the solvers become more and more efficient. Amazing episode, Jerry. I learned so much from you yet again. Do you have a book recommendation for us again this time?

Jerry Yurchisin: 01:34:27 Sure. Last time I dropped some, hey, let's get into optimization as one of the books I cheated and used too. This one's going to be a little bit different. I am a child of the '80s and '90s, so one of the big things that a lot of people had in the '80s and '90s was some sort of Nintendo system. So the book that I'm going to highlight now is called, Ask Iwata. So Satoru Iwata was a legendary CEO at Nintendo, was given a lot. He's sort of given a lot of credit for just fostering a very creative environment that

allowed developers and people that work there to really push the boundaries of imagination and things like that in game development. So it sort of goes into his leadership and management philosophy, so sort of emphasizing things like empathy for your workers and customers and really trying, really putting your heart and soul into what you're trying to create. And things like innovation, risk taking and how that's very important for companies to thrive.

01:35:54 It's just something that I found to be extremely interesting and something that I sort of took to heart a little bit in terms of risk-taking, particularly it's kind of just like coming to a position like this at Gurobi was a little... I could keep working in consultancy doing a lot of cool projects and stuff like that, but being able to take a risk and start really getting a message out to data scientists and the AI community about optimization, I thought that was a big risk. It could have gone nowhere and I could have not be on podcasts like this talking to awesome people like you and everything. Like that could have just tanked and then I'd be back to where I was. I sort of took that to heart. So I think that could be a great thing for anyone else who's interested in the history of Nintendo and things like that could be interested to dive into that book.

Jon Krohn: 01:36:51 Great message there. And it's funny, I'm also a child of the '80s and '90s. I remember one of my earliest memories, I must have been four years old on a bus. I remember I was leaving kindergarten to go home on a bus and I was sitting next to some kid, I have no idea who he was, but he talked about how soon there was going to be a Super Nintendo and it blew my mind and that might literally be my earliest memory [inaudible 01:37:20].

Jerry Yurchisin: 01:37:20 That's a good one, that's a good one to have. If you're going to keep one, that is some of my first memories is I

have a older brother, if you have older brothers, they tend to pick on you a little bit and beat you in things mercilessly and getting beaten in video games is one of my earliest memories. I don't know if he's going to ever hear this or anything, but I'm significantly better than him now for a long time. I have significantly outpaced him. But no, it's all in good fun and those types of fun playing games with people and stuff like that is something I look very fondly back on and definitely miss those times a little bit hanging out on the couch and playing games together is one of my favorite past times.

Jon Krohn: 01:38:13 Yeah, I do miss when we were all kids and everyone just had all this time and you could just phone a friend up and hang out and you just knew that they were at home just looking for something to do and you don't have that as an adult. I guess we have retirement to look forward to.

Jerry Yurchisin: 01:38:30 That is true.

Jon Krohn: 01:38:32 In the meantime, between listening to this episode and retirement, Jerry, where should people be following you to get your latest thoughts?

Jerry Yurchisin: 01:38:40 Sure. Probably LinkedIn is probably the best way. That's where I'm most active, because a lot of the stuff that I put out there is also through Gurobi as well, and you can stay connected with all of the coolest advancements that I was talking about with non-linear capabilities and all this other stuff. You can definitely find out all of our awesome events. If you want to go to Las Vegas, if you're looking for a reason to go to Las Vegas, we are holding a summit in mid-September. It's going to be great. I'm going to be hosting a data science track so we can sit down together and have fun with all these hands on stuff if you're interested in that. And you'll be getting a lot of updates through LinkedIn on that. I'm also active, somewhat

		active on Threads, but by and large, LinkedIn's the way to get to me or just email me. That's cool too or LinkedIn message. I'm happy to chat with anyone who's interested.
Jon Krohn:	01:39:47	Nice. And yeah, I just checked out the Gurobi Summit coming up September 19th and 20th at not just anywhere in Las Vegas, but the Wynn Encore Resort, which is as far as I'm aware, the premier spot. It's the only place I've stayed in Vegas and it was a really cool, beautiful spot.
Jerry Yurchisin:	01:40:06	Awesome. Yeah, we're super excited about the whole event and I was talking about NVIDIA, they're going to be coming to speak there as well, so you can hear more about the relationship between Gurobi, mathematical optimization and the powerhouse known as NVIDIA.
Jon Krohn:	01:40:22	Nice. And actually registration is about a 10th of what it usually would be. Registration is $200 to $300 depending on when you sign up before or after September 3rd. But typically I'd expect to be going to a conference in Vegas at the Wynn Encore, it'd be about 10 x that. So yeah, it does look like a great excuse and you wouldn't have to bug anyone at your company for too much budget to head out there and check it out. Very nice.
Jerry Yurchisin:	01:40:55	Cool.
Jon Krohn:	01:40:55	All right, thanks, Jerry. Thank you so much for taking all the time with us today. It's been awesome. I have learned a ton and yeah, can't wait until the next time.
Jerry Yurchisin:	01:41:03	Awesome. Sounds good, Jon. I really appreciate it and yeah, looking forward to more conversations.
Jon Krohn:	01:41:14	What a brilliant, well-spoken guy. In today's episode, Jerry filled us in on how mathematical optimization is prescriptive, such as helping with business decisions relative to ML and stats predictive nature. And he talked

about how mathematical optimization is the ideal tool for the job whenever there are many real world constraints to factor in and you'd like to maximize or minimize something. Talked about how you can learn about optimization hands on yourself using his Jupiter notebooks and online courses. We've got links to all of those in the show notes. He talked about how GPUs are not ideal for optimization, but state-of-the-art CPU's, like the 72 core NVIDIA GH200, allow optimization operations to run 23% faster and use nearly half as much energy. He talked about how the latest and greatest mathematical optimizers can handle quadratic inputs and outputs and how NP-hard problems like the traveling salesman problem and the knapsack problem can now in some cases be handled in seconds by mathematical optimizers while just a few years ago we might not have even attempted to tackle such complex problems.

01:42:19 As always, you can get all the show notes including the transcript for this episode, the video recording, any materials measured on the show, the URLs for Jerry's social media profiles, as well as my own at superdatascience.com/813. Thanks of course to everyone on the Super Data Science podcast team, our podcast manager, Ivana Zibert, media editor Mario Pombo, operations manager Natalie Ziajski, researcher Serg Masis, writers Dr. Zara Karschay and Silvia Ogweng, and founder Kirill Eremenko for producing another outstanding episode for us today.

01:42:48 We're enabling that super team to create this free podcast for you. We are very grateful indeed to our sponsors. You can support this show by checking out our sponsors links which are in the show notes. And if you are interested in sponsoring an episode, you can get the details on how by making your way to jonkrohn.com/podcast. Otherwise, please share, review, subscribe, all that good stuff. But most importantly, just keep on tuning in. I'm so grateful

to have you listening and hope I can continue to make episodes you love for years and years to come. Till next time, keep on rocking it out there and I'm looking forward to enjoying another round of the Super Data Science podcast with you very soon.