Six Life Lessons You Didn’t Learn from your OR Program

Thank you for joining us. We will be starting shortly.
Welcome to the Webinar

Six Life Lessons You Didn’t Learn from your OR Program
Speaker Introduction

Dr. Michael Watson

- CEO at Opex Analytics
- Recognized leader in analytics and supply chain optimization
- Michael was an early employee and officer at LogicTools
- While at IBM, he was the worldwide business leader for network design, inventory and routing solutions
- Co-author of “Managerial Analytics” and “Supply Chain Network Design”
- Adjunct professor at Northwestern University teaching graduate level courses within the Master in Engineering Management and the Master of Science in Analytics programs
Speaker Introduction

Dr. Pete Cacioppi

• Algorithm Consultant at Opex Analytics

• First employee and Chief Scientist at LogicTools

• Co-founder and Chief Scientist at Opalytics (now part of Accenture)

• Co-author of “Supply Chain Network Design”

• Decades of experience building robust optimization applications with multiple programming languages.
Six Life Lessons You Didn’t Learn from your OR Program

These are our lessons, what we have seen

OR programs teach you OR (as they should)

These lessons are about getting your OR code used

We’ll bring a CS and Business element
Not our first lesson

Users Like Nice Interfaces

We aren’t going to waste one of our six on this
Pay Attention to the End User

“If we want users to like our software, we should design it to behave like a likeable person: respectful, generous and helpful.”

Alan Cooper
Software Designer and Programmer
For the Business User

You Need Input (Manual and Automatic)

<table>
<thead>
<tr>
<th>Customer</th>
<th>Material</th>
<th>Time Period</th>
<th>Demand (Cases)</th>
<th>Minimum Demand To Be Met</th>
<th>Unmet Demand Penalty ($ per Case)</th>
<th>Meta Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>24</td>
<td>13</td>
<td>2</td>
<td>1.920.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>13</td>
<td>2</td>
<td>4.683.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>13</td>
<td>2</td>
<td>4.032.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>13</td>
<td>2</td>
<td>1.728.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>13</td>
<td>2</td>
<td>3.640.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>13</td>
<td>1</td>
<td>6.012.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>13</td>
<td>2</td>
<td>50.112.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>13</td>
<td>3</td>
<td>2.880.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>13</td>
<td>4</td>
<td>24.260.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>13</td>
<td>5</td>
<td>8.736.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>13</td>
<td>6</td>
<td>26.880.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>13</td>
<td>1</td>
<td>14.976.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>13</td>
<td>2</td>
<td>27.840.00</td>
<td>0.00</td>
<td>100.00</td>
<td></td>
</tr>
</tbody>
</table>
For the Business User

Users Like to Have Control over Parameters
For the Business User

And, they like reports to see what happened
For Consumer or Embedded Systems

The interface might not be your problem

But the principles of “respectful, generous and helpful” still apply (both to the end user and to your fellow engineers).
When Building an OR Interface, Less can be More
When Building an OR Interface, Less can be More
Don’t Be Hard On Yourself

Your first version won’t be that good
Proof of new analytics concept
2 countries
5-10 users
Monthly

Production-level system
49 countries
200-300 users
Twice-daily

Don’t Be Hard On Yourself
Your first version won’t be that good
When We Talk Interface, we mean...

- GUI
- Input
- Control LP/MIP
- Output
Lesson #1
Don’t Let Users Get Themselves In Trouble

Their trouble becomes your trouble
Story Told by Dr. Jeffrey Karrenbauer
(as a Guest Lecturer at Northwestern in the mid 1990’s)
He designed a system for a client that had a parameter limited to a single digit.

I don’t remember what this parameter controlled, but it had something to do with a limit on some very hard variables.
The client had filled in a value of 9.

But, the client begged them to release the limit just so they put in a value of 10 or 11, maybe 12 at the most.
So, they did, they gave them a 2-digit entry here…..

… and soon the client was calling because the model failed.

… they spent a lot of time trying to figure it out….

Finally they looked at this parameter
Story Told by Dr. Jeffrey Karrenbauer
(as a Guest Lecturer at Northwestern in the mid 1990’s)

Parameter X (1-9 are valid)
Story Told by Dr. Jeffrey Karrenbauer
(as a Guest Lecturer at Northwestern in the mid 1990’s)

Parameter X (1-9 are valid) 99

The client had taken this parameter to 99.

Of course, it was limited to 9 for a reason.
Story holds today, just in different forms

If you are using a Set Covering approach

(For vehicle routing or cutting stock)
Story holds today, just in different forms

If you are using a Set Covering approach

(For vehicle routing or cutting stock)

It is easy to imagine that some average cases have a tractable number of feasible inputs (say limits of 5 stops, or limits of 3 cuts in 5 inch increments).
Story holds today, just in different forms

If you are using a Set Covering approach

(For vehicle routing or cutting stock)

It is easy to imagine that some average cases have a tractable number of feasible inputs (say limits of 5 stops, or limits of 3 cuts in 5 inch increments).

It is also easy to imagine giving your users some control of the parameters and then accidentally unlocking choices with combinations that would take weeks (or longer) to generate.
Today Gurobi’s MIP Solver is Awesome

It can solve large problems

It can model almost anything
Today Gurobi’s MIP Solver is Awesome

It can solve large problems

It can model almost anything

But, still, $P \neq NP$
Gurobi Power and Flexibility + P ≠ NP:

Shows up in design….

“Can I model all my products?”

- ‘Yes, Gurobi can handle large models’
Gurobi Power and Flexibility + P ≠ NP:

Shows up in design....

“Can I model 52 time periods?”

- ‘Yes, multiple time periods are quite common
“Can I do open/close (use/not use) decision at lots of locations?”

- ‘Yes, MIPs are always getting faster
Gurobi Power and Flexibility + P ≠ NP:

Shows up in design....

“Can I model fixed set up and switching?”
- ‘Yes, there are ways to do this
Gurobi Power and Flexibility + $P \neq NP$:

Shows up in design....

“Can I ....”

- Probably, yes....
Gurobi Power and Flexibility + \( P \neq NP \):

Shows up in design….

“Can I model all my products?”
- ‘Yes, Gurobi can handle large models’

“Can I model 52 time periods?”
- ‘Yes, multiple time periods are quite common

“Can I do open/close (use/not use) decision at lots of locations?”
- ‘Yes, MIPs are always getting faster

“Can I model fixed set up and switching?”
- ‘Yes, there are ways to do this

“Can I ….”
- Probably, yes….

Each one of these is perfectly reasonable and we’ve seen models that do them….
"Can I model all my products?"
- ‘Yes, Gurobi can handle large models’

"Can I model 52 time periods?"
- ‘Yes, multiple time periods are quite common’

"Can I do open/close (use/not use) decision at lots of locations?"
- ‘Yes, MIPs are always getting faster

"Can I model fixed set up and switching?"
- ‘Yes, there are ways to do this

"Can I ...."
- Probably, yes....
What do you need to do about all of this?

1. **Set up guardrails:** Don’t let users automatically kick off near-infinite loops of input or create a large set of integer variables that will certainly fail.

2. **Build Heuristics:** guardrails might not work. Build a heuristic that builds and solves a smaller problem.

3. **Educate:** teach your users good modeling so they build good models that solve and answer the question.
What do you need to do about all of this?

My least favorite one (because I always lost)…
What do you need to do about all of this?

My least favorite one (because I always lost)...

We were responsible for strategic facility location tool

Me: Users really wanted to model each truck or at least step sizes on lane
Pete: Even if we gave them the feature, they wouldn’t actually be able to use it!!
What do you need to do about all of this?

My least favorite one (because I always lost)…
Lesson #2, #3 and #4

THE 3 P’S

Be Prepared. Be Proactive. Be Paranoid
Lesson #2

Be Prepared (Not Proactive) to Become Fast

Speed is less important (and less controllable) than you think.
Classic Trope of Computer Science:

90/10 (or 80/20) profile for slow programs

Wait for real data

You can’t predict the bottleneck
Don’t Slow Down For Speed

Premature optimization is the root of all evil.

— Donald Knuth —

Optimization = Fast Code (in this case)
1. Know how to profile your engine with reality based performance complaints (cProfile works well with Python)

2. Know how to “slice” with whatever language you’re using (tuplelist with Python)
At some point, you will have addressed the bottlenecks

A MAN'S GOT TO KNOW

HIS LIMITATIONS
At some point, you will be removed obvious bottlenecks

### Table: Efficiency Comparison

<table>
<thead>
<tr>
<th></th>
<th>Case 1</th>
<th>Other Cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating Input</td>
<td>33%</td>
<td>90%</td>
</tr>
<tr>
<td>Running Solver</td>
<td>33%</td>
<td>90%</td>
</tr>
<tr>
<td>Creating Output</td>
<td>33%</td>
<td>90%</td>
</tr>
</tbody>
</table>

Likely Removed it

If you’ve worked, you may be stuck
don’t let PERFECT be the enemy of the GOOD.
Lesson #3
Be Proactive with Bugs

Strive for Zero Defect Software
Do Slow Down to Catch Bugs  or  Go Slow to Go Fast

Bugs not only erode credibility, they waste time.

Don’t release untested software.

Any released bugs should be post-mortem-ed for lessons learned. Rhinoceros become Jedis.
Do Slow Down to Catch Bugs

Put the O-Rings in ice water

You’re going to write fewer math modeling bugs than you expect, but they can be incredibly insidious.
Do Slow Down to Catch Bugs

Write code to test code.

• Don’t let testing be “tears in rain”.
• Unit test and Jenkins are perfect fits for Gurobi engines…. Why aren’t more people doing this?
Do Slow Down to Catch Bugs

False Bugs (users that want magic) aren’t counted in “zero defect” scoring.
Some clients don’t know what they want and correct code will still "feel wrong" to them. This is to be expected, and zero defect development is even more important for such engagements.
Lesson #4
Be Paranoid About Solver Failures

Users hate infeasibility but they appreciate insights
A solve that fails with a cryptic message is always unwelcome. A good engine always has something helpful to say.

Two types of solve failures

1. Dirty data
2. Infeasible models
Dirty Data Should be Handled Proactively

Protect your fragile math model with rich integrity checks

Build the GUI to prevent dirty data from entering (user cleans it outside)

Build the GUI to allow the user to clean the data in the GUI (but not run yet)

(Bad Choice): Just assume that clean data is obvious and let it go
Infeasible model support is a blend of proactive and prepared

- Slack variables (with penalties) can be useful, but only if the user understands the results
- “This-might-create-an-infeasibility” data checks are best added in response to real world complaints. You should be prepared on how to generate these supplemental reports without drowning out solution reports

Add to a knowledge base over time of errors and problems
Lesson #5
You Will Be Swimming in Data

Don’t forget the data in data science
Data is the first word in data science

You will spend more time working with data than with math.
  • The MIP engine is really just one more tool that generates insight about data.
  • Need to have expertise with tools that let you study data quickly and efficiently

Prefers “Raw Python”
Gets pandas, SQL

Old school Excel,
Consider Alteryx, or a visualization tool
Data is the first word in data science

Part of swimming in data is about making it easier to read that data

Following two tricks can drastically simplify .ilp files for infeasible model support

- ‘Obfusimplify` trick for simpler .lp/.ilp files - makes infeasible models even easier.
- Model shrinking with cascading deletes - also makes infeasible models easier.
Don’t Let Design and Build Get Ahead of Data

Don’t let this happen:

You want to build a model with really detailed features (like details at the workstation line, or down to the worker skill). You build such a model…

You are never able to get the data to feed the model (you can’t get workstation data, you can’t get worker skills).
Lesson #6
Take advantage of modern tools and don’t be afraid to migrate to new ones

Not easy, but we did it many times and will do so again
Switching Technology is Painful...

We’ve done it many times since 1997:
- MapInfo to VB and Access to … to .Net to Cloud Based….

It was painful every time—no new features, just new platform

It almost always seemed to pay off!
The Tools Matter


“Consultants”
• Used Excel and Access
• Did “pivot tables” and “cross tab queries”

“Developers”
• Used low level code in compiled languages
• Worried about memory management

Both use Python (or R)
• Share info, share code
• Do it yourself coding, developers can take notebooks from consultants
The Tools Matter


“Consultants”
• Used Excel and Access
• Did “pivot tables” and “cross tab queries”

“Developers”
• Used low level code in compiled languages
• Worried about memory management

Python has momentum
• Big and growing ecosystem
• Still need to worry about questions like “has Library Y made Library X obsolete?”
• And, still need to watch out for the next generation of tools
• Don’t stay stagnant!
Thank You – Questions?
Your Next Steps

- If you haven’t already done so, please register for an account at http://www.gurobi.com
- For questions about Gurobi pricing contact sales@gurobi.com or sales@gurobi.de
- A recording of this webinar, including the slides, will be available in roughly one week
- Please access https://support.gurobi.com with any technical questions