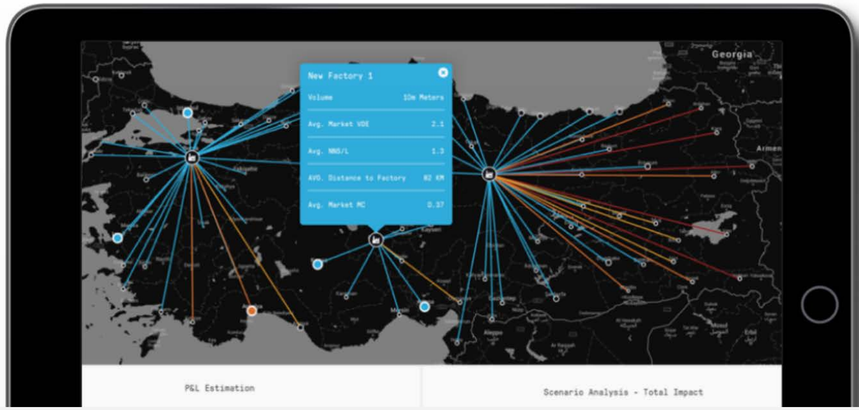
USING PULP AND SOLVERS FOR BUSINESS ANALYTICS

NOVEMBER 29, 2019

MOKA

At MOKA, we build intelligent planning software that allows organizations to make smarter, faster, strategic decisions.



Unifies strategic data model, automates & speeds analysis, and addresses data gaps

Delivers a **lasting organizational capability.** Enterprise-ready, secure & light-weight for fast deployment

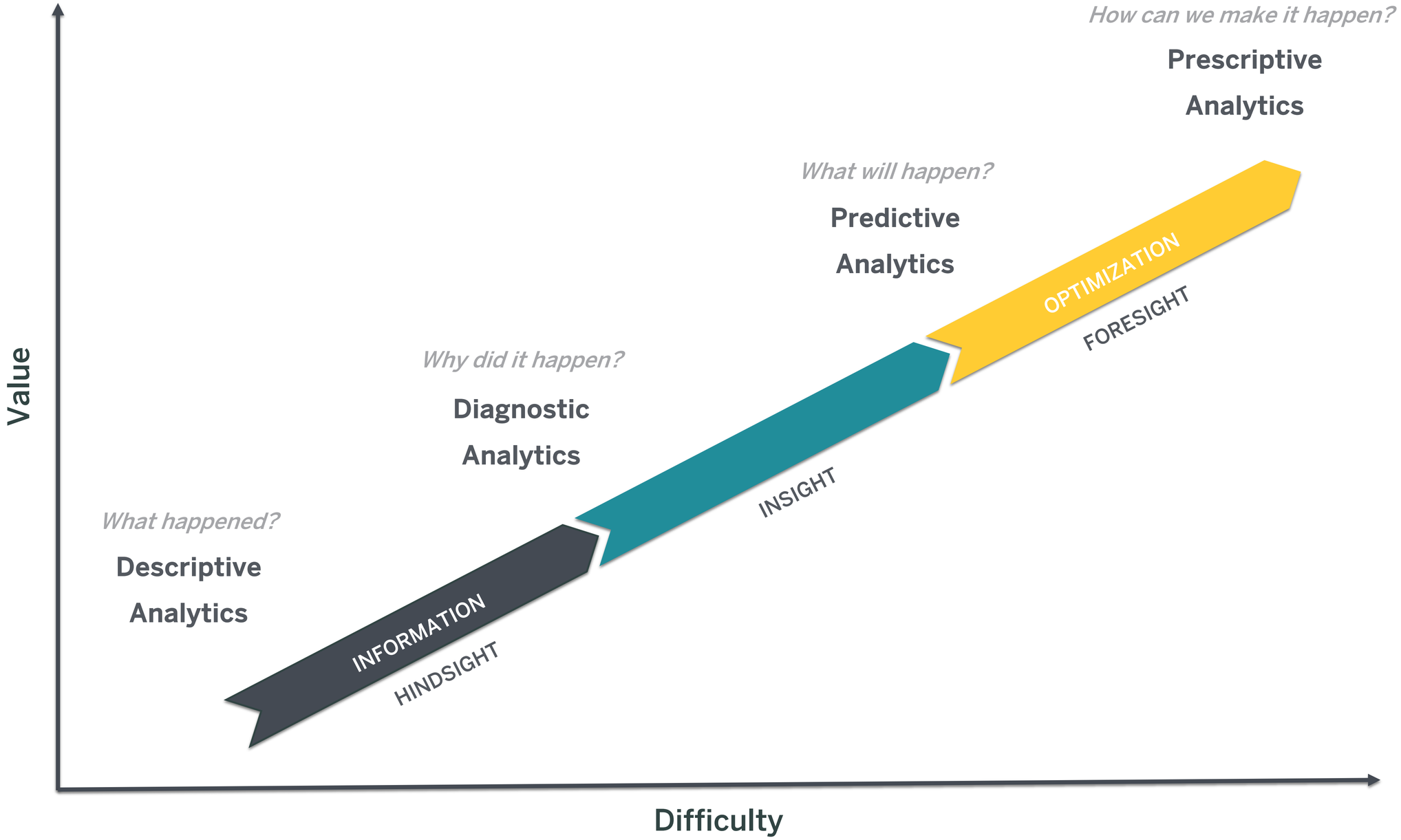
Predictive & prescriptive analytics drive insights & evaluate complex scenarios

Encode and distribute **high-quality strategy frameworks** across your entire organization

The Winning Model powers strategic planning for manufacturing business models – from CPG to automotive to industrials.

Ready to go out-of-the box, we tailor the core modules and intelligence engine to a client's specific business.

BUSINESS ANALYTICS VALUE LADDER



Source: Gartner

Talk Outline

1

What is Linear Programming?

2

Using PuLP for Macro Capacity Planning

3

Python Solver Landscape

4

Best Practices for Using Solvers for Analytics

LINEAR PROGRAMMING

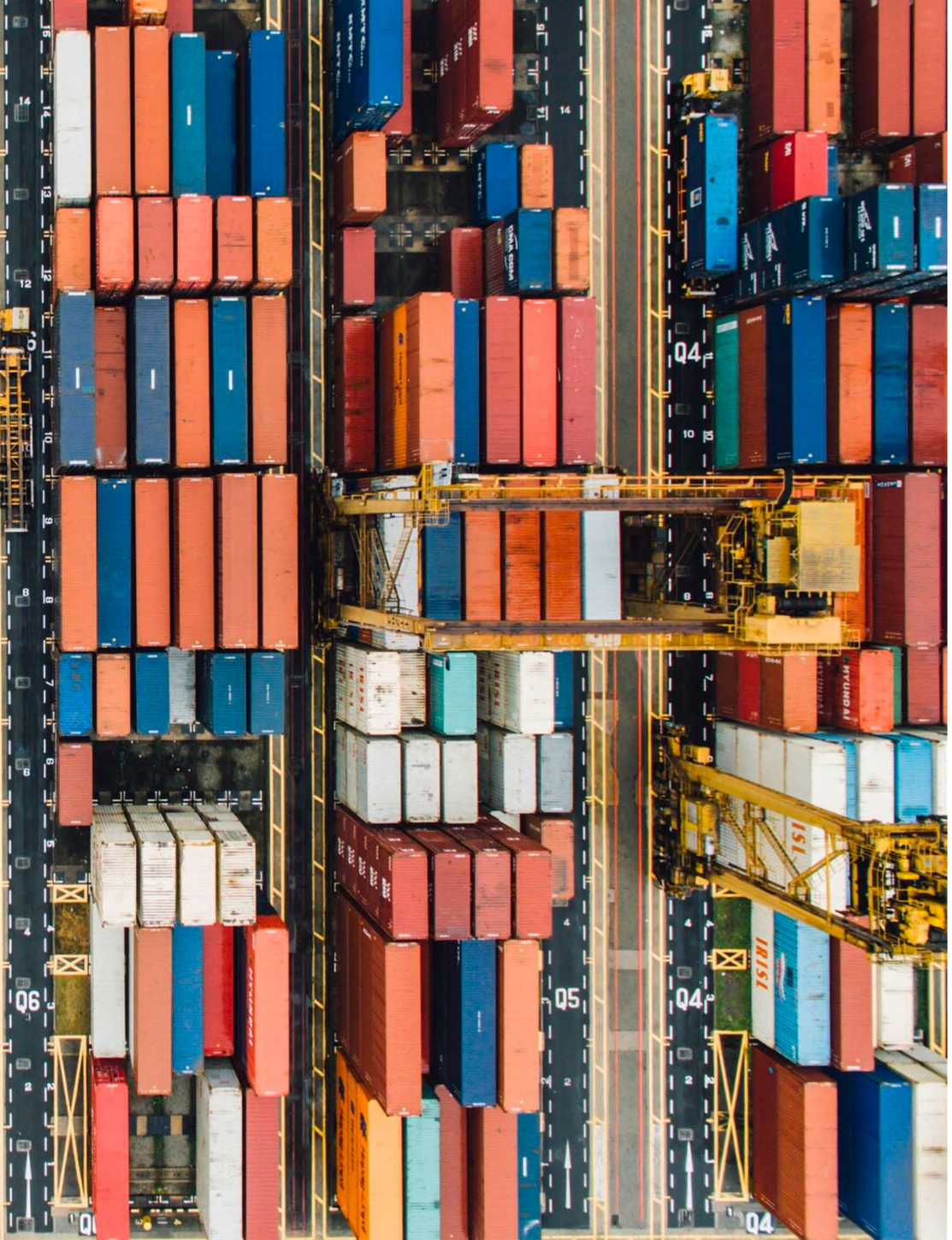
	Matrix Form	Standard Form
minimize	$\mathbf{c}^{\mathrm{T}}\mathbf{x}$	$f(x_1, x_2, \ldots) = c_1 x_1 + c_2 x_2 + \cdots$
subject to	$A\mathbf{x} \leq \mathbf{b}$	$a_{11}x_1 + a_{12}x_2 + \cdots \leq b_1$ $a_{21}x_1 + a_{22}x_2 + \cdots \leq b_2$ $a_{31}x_1 + a_{32}x_2 + \cdots \leq b_3$ …
and	$\mathbf{x} \geq \mathbf{0}$	$x_1 \geq 0$ $x_2 \geq 0$ …

Trick #1: negate all the constants to maximize a function

Trick #2: re-arrange expressions (e.g., subtract from both sides) to isolate constant on the right side

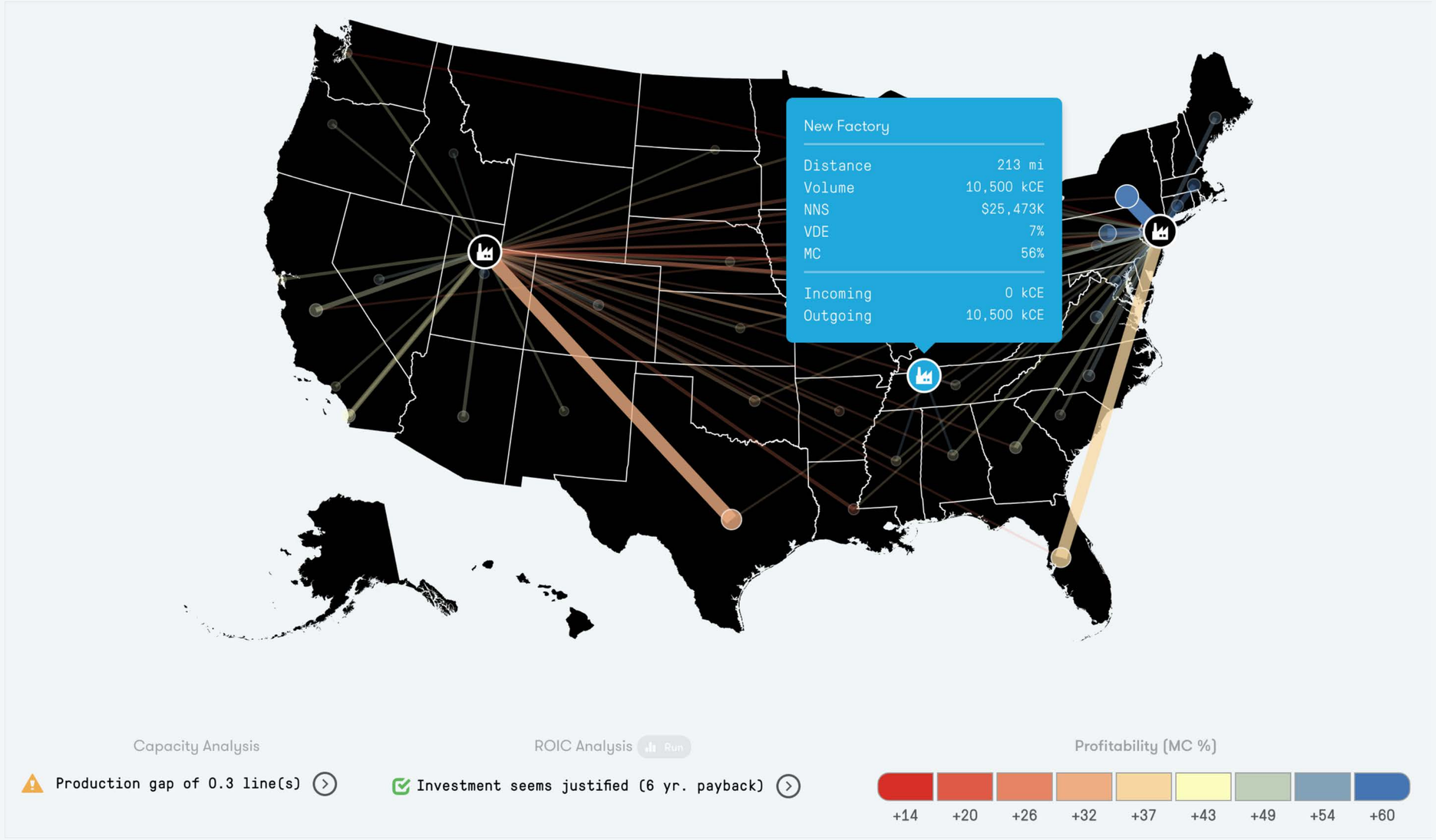
Trick #3: add slack/dummy variables to encode other relationships, e.g., strict equality

Many tools, including PuLP, can apply these tricks automatically to allow a more natural expression



MACRO CAPACITY PLANNING: BUSINESS QUESTIONS

- Where are the inefficiencies in our supply chain?
- Will we have enough production capacity if demand grows?
- Is our supply chain positioned to profitably capture growth opportunities?
- Should we open a new warehouse and/or factory? If so, where?



With the Winning Model's Network Strategy Module, Network Planners and CEOs can evaluate network strategy and investment opportunities instantaneously

Jupyter Walkthrough

https://github.com/moka-analytics/engineering

PYTHON SOLVER LANDSCAPE

[1] Not all paradigms shown. Wikipedia is a good resource for exploring various paradigms

Paradigm[1]	Problem Form	Example Use Cases	Example Python Packages
Mathematical/Numerical			
Linear Programming	$\begin{aligned} &\text{minimize} && \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b} \\ &\text{and} && \mathbf{x} \geq \mathbf{0} \end{aligned}$	• Supply chain optimization • Production planning	• PuLP: interface to linear and mixed-integer solvers • MIPCL: commercial mixed-integer programming • GLOP: Google's LP-only solver
Integer Programming	$\begin{aligned} &\text{maximize} && \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}, \\ &\text{and} && \mathbf{x} \in \mathbb{Z}^n, \end{aligned}$	• Minimize interference across cellular network • Bus scheduling/Vehicle routing • "Knapsack" problem	
Quadratic Programming	$\begin{aligned} &\text{minimize} && \tfrac{1}{2}\mathbf{x}^{\mathrm{T}}Q\mathbf{x} + \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ &\text{subject to} && A\mathbf{x} \leq \mathbf{b}, \end{aligned}$	• Financial portfolio optimization • Image/signals processing • Least-Squares regression	• qpsolvers: unified interface around quadratic solvers • quadprog: implementation of the Goldfarb/Idnani dual algorithm
Convex Optimization	$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) \leq 0, \quad i = 1, \ldots, m \end{aligned}$ $f, g_1 \ldots g_m : \mathbb{R}^n \to \mathbb{R}$ are all convex	• Training ML models • Linear/Quadratic are a special case of Convex	• cvxpy • cvxopt
Non-Linear Programming	$\begin{aligned} &\text{minimize } f(x) \\ &\text{subject to } g_i(x) \leq 0 \text{ for each } i \in \{1, \ldots, m\} \\ &\qquad h_j(x) = 0 \text{ for each } j \in \{1, \ldots, p\} \\ &\qquad x \in X. \end{aligned}$		• pyOpt
Multi-Paradigm			• Pyomo: multi-paradigm interface to multiple solvers • Google OR-Tools: Google's operations research tools • Gurobi: commercial optimizer supporting multiple languages • SciPy: its optimize package contains numerous solvers
Logical/Constraint			
Logic Programming	P(x). ?- Q(x).	• Expert system AI • Knowledge graph inference • Natural Language Processing (NLP)	• logpy (aka kanren) • PySWIP: SWI-Prolog bridge • ErgoAI: via a Java bridge • Pyke: pure Python
Satisfiability Modulo Theories (SMT)	$\exists x. P(x)$	• Static software analysis • Program synthesis • Scheduling	• pySMT: interface to SMT-LIB format supporting all major solvers • Z3: Microsoft's SMT solver with support for multiple • CP-SAT: Google's constraint programming solver

BEST PRACTICES FOR USING SOLVERS FOR ANALYTICS

Avoid false precision

Know when to tradeoff speed vs. precision

Educate users about what's being optimized, e.g., assumptions and limitations

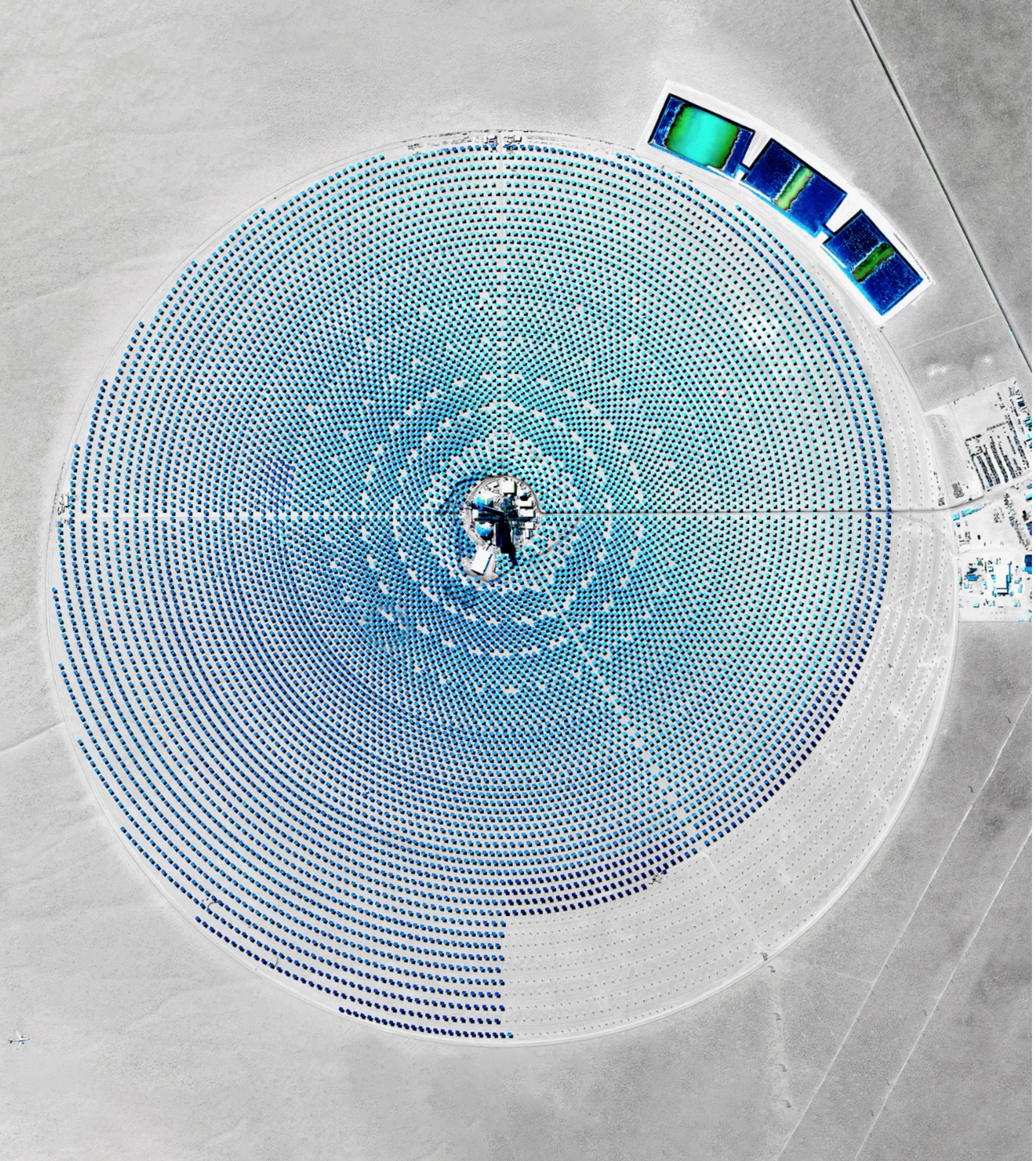
Support user intuition with analytics and visualizations

Don't be afraid to run a solver multiple times

Get creative with elastic computing resources

Benchmark optimizers to find best for your workload

Profile your Python problem creation code



KEY TAKEAWAYS

- Important business problems can be framed as optimization problems
- There’s a Python interface for every kind of problem
- Augment solver output with complementary analyses and visualizations

We’re Hiring!

- Senior Full Stack Engineer
- Senior Front-End Engineer
- Data Integration Engineer
- Vice President, Deployment Strategy
- Business Project Manager

Come find me after the talk, or email me at todd@moka.nyc

We offer $$$ for successful referrals

Slides and notebook available on GitHub:

https://github.com/moka-analytics/engineering

Learn more about MOKA Analytics at

https://www.moka.nyc

Find more of my talks and articles at

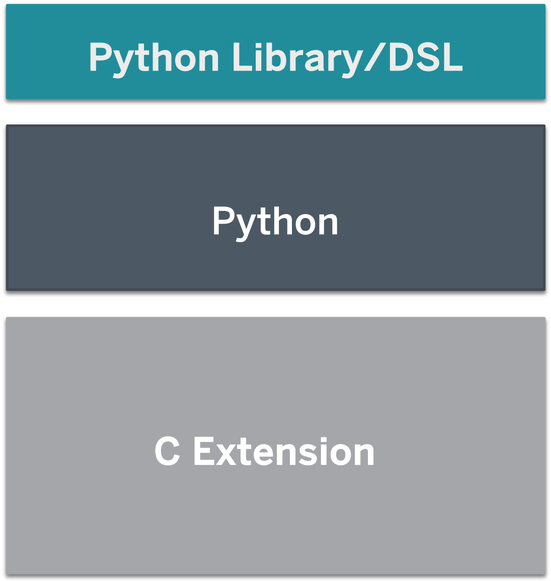
https://toddschiller.com/

Appendix

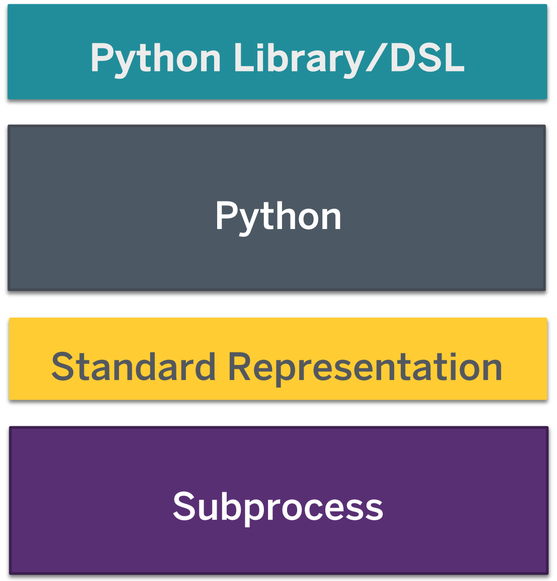
THREE PYTHON SOLVER ARCHITECTURES

Pure Python

Python w/ C



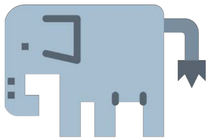
Python w/ External Executable



Can easily swap solvers. But pay a speed penalty for (de)serialization

DSL = Domain Specific Language

TACTICS FOR SUPPORTING INTUITION BEHIND OPTIMIZATION RESULTS



Focus on major drivers / points of sensitivity

Show local trade-offs

Leverage existing descriptive and diagnostic analytics

Show “diff” of output as user tweaks assumptions