Modeling, not Programming

Model-Based Optimization in AMPL

Robert Fourer
4er@ampl.com

AMPL Optimization Inc.
www.ampl.com — +1 773-336-2675

Technology Tutorial
INFORMS Virtual Annual Meeting
10 November 2020

Optimization in Analytics

Goals

Features

Applications

Steps

Optimization *Goals*

Given a recurring need to make many interrelated decisions

- Purchases, production and shipment amounts, assignments, . . .

Consistently make highly desirable choices

By applying concepts of mathematical optimization

- Ways of describing problems *(formulations)*
- Ways of solving problems *(algorithms)*

Optimization *Features*

Large numbers of decision variables

- *Thousands to millions*

An objective function

- *Minimize or maximize*

Various constraint types

- *10-20 distinct types, though large numbers of each type*
- *Few variables involved in each constraint*

Numerous scenarios with different data

- *Can't characterize all possible solutions in advance*

Optimization *Applications*

Energy and Utilities

- power networks, gas pipelines, hydroelectric power, water distribution

Production

- mining, steel, chemicals, oil refining, forestry and paper
- cars & trucks, paper products, processed foods

Transportation

- airlines, trucking, package delivery

Services

- supply chain, hospitals & medicine, construction management

Communications

- telecommunications, professional networking, file hosting

Finance

- software tools, investment management, commodity management

Advanced Technologies

- artificial intelligence, distributed computing, biotechnology

Optimization *Development Steps*

Communicate with Client

↓

Build Model or Method

↓

Prepare Scenarios

↓

Solve Cases

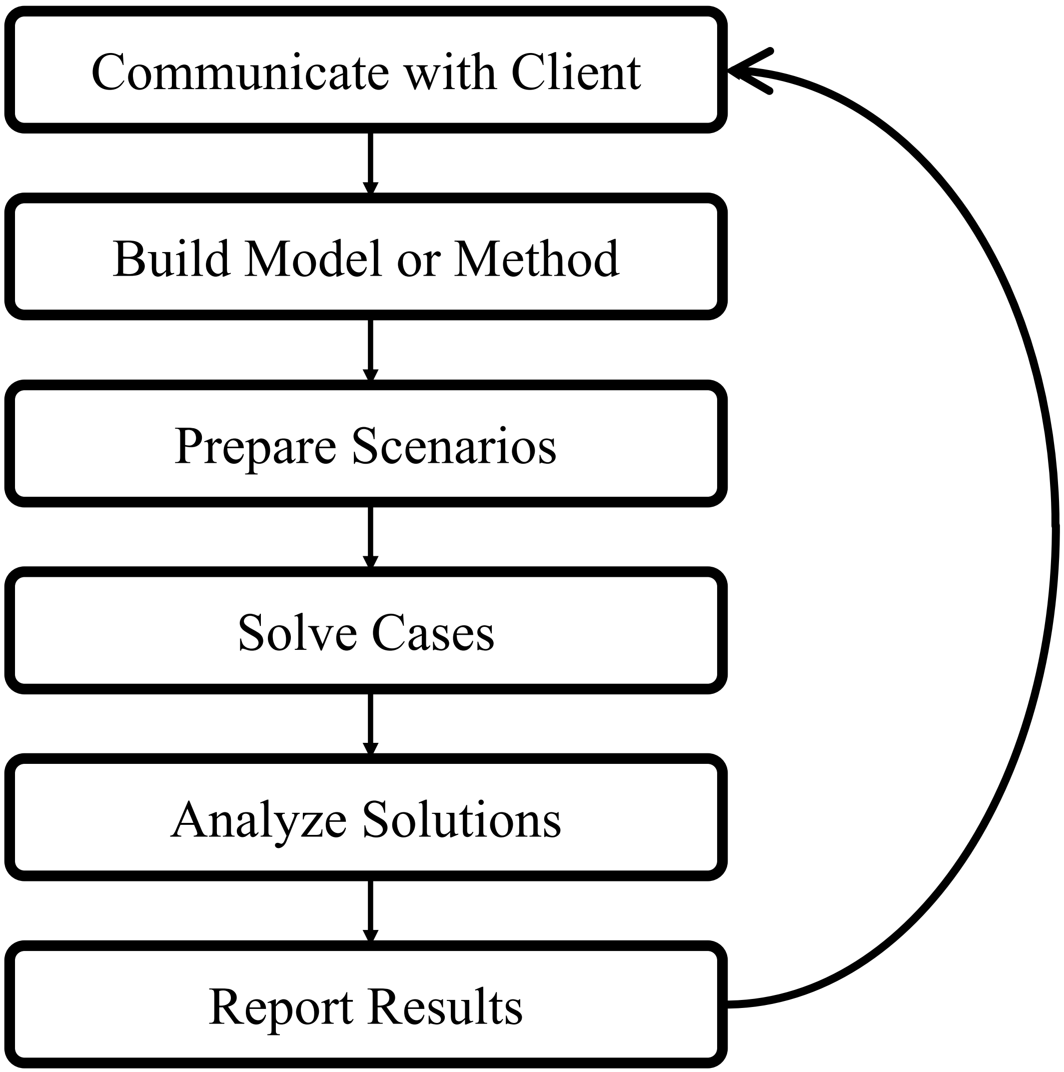
↓

Analyze Solutions

↓

Report Results

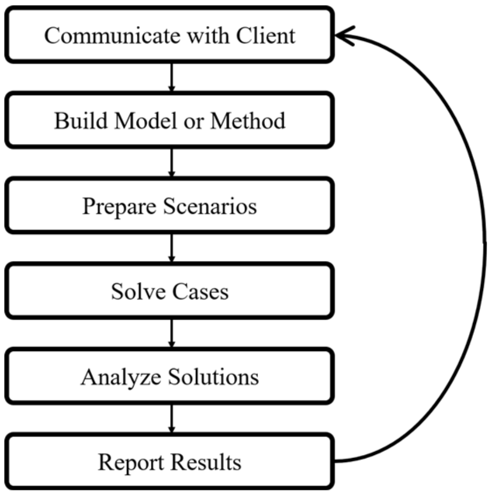
Optimization *Development Cycle*



Optimization *Development Cycle*

Goals for optimization practitioners

- Repeat the cycle quickly and *reliably*
  - Get results before client loses interest
- Deploy *effectively* for application
- Update as needed



Goals for optimization software

- Promote fast prototyping
- Facilitate integration with application systems
- Encourage long-term maintenance

Overview

Modeling, not programming

Comparison of approaches

- Optimization: *Model-based* or *method-based*?
- Model-based optimization:
  Modeling language or *programming language*?
- Modeling languages: *Declarative* or *executable*?

Case studies

- Packing shipments
- Designing aircraft
- Managing power grids
- Assigning students to classes

Approaches to Optimization

Method-based approach

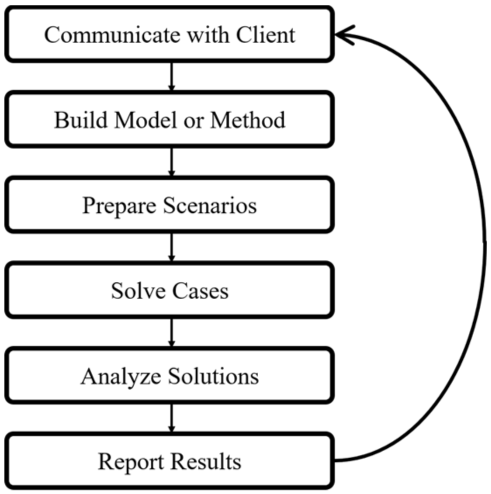
- *Program* a method (algorithm) for computing solutions

Model-based approach

- *Formulate* a description (model) of the desired solutions

Which should you prefer?

- For simple problems, any approach can work
- But the application development cycle introduces *complications . . .*



Example:

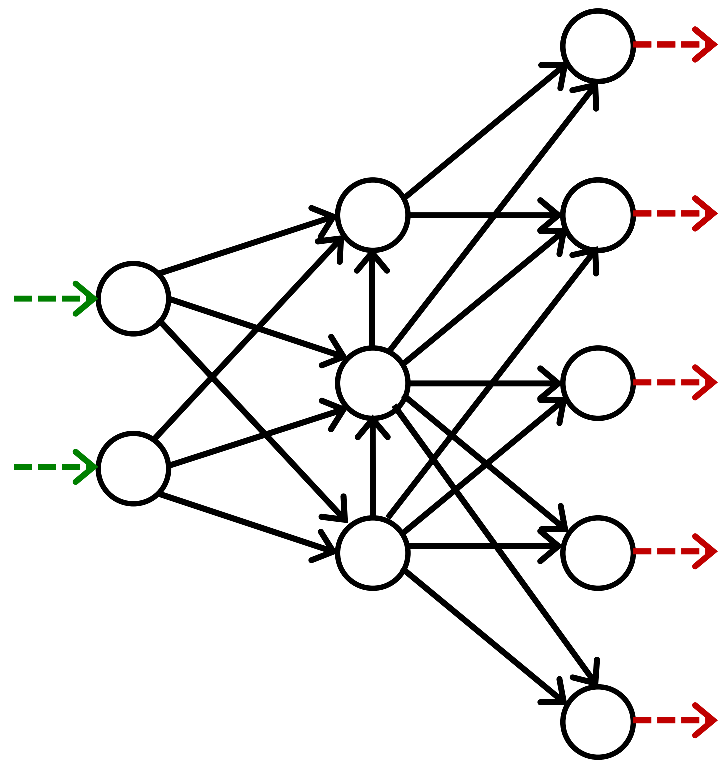
Supply Chain Optimization

Motivation

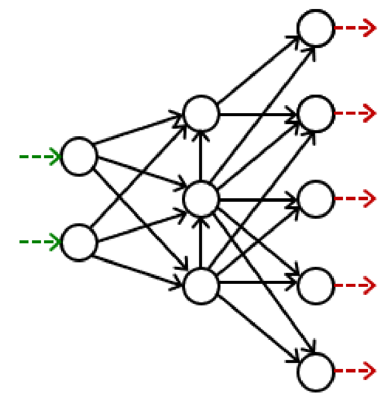
- Ship products efficiently to meet demands

Context

- a transportation network
  - locations
  - links
- supplies at locations
- demands at locations
- capacities on links
- shipping costs on links



Supply Chain Optimization



Decide

- how much of each product to ship on each link

So that

- shipping costs are kept low
- shipments on each link respect capacity of the link
- supplies, demands, and shipments are in balance at each location

Two approaches . . .

Method-Based Approach

Program a method to build a shipping plan

- *method:* says how to compute a solution

Order-driven

- Develop rules for how each order should be met
  - Given some demand and given available capacity, determine where to ship it from and which route to use
- Fill orders one by one, according to the rules
  - Decrement capacity as each one is filled

Route-driven

- Repeat until all demands are met
  - Choose a shipping route and a product
  - Add as much flow as possible of that product along that route without exceeding supply, demand, or capacity

Method-Based Approach

Program refinements to the method to get better results . . .

Enhance the method

- Fill large order first, *or*
- Consider the least expensive routes first

Improve the initial solution

- Look for simple exchanges that reduce cost

Apply metaheuristic concepts

- Systematically search for local improvements
  - simulated annealing, tabu search, GRASP
- Combine solutions to evolve better ones
  - evolutionary methods, particle swarm optimization

. . . usually no optimal method is available

Model-Based Approach

Formulate a minimum shipping cost model

- ***model:*** says what a solution should satisfy
- Identify amounts shipped
  as the decisions of the model *(variables)*
- Specify feasible shipment amounts
  by writing equations that the variables must satisfy *(constraints)*
- Write total shipping cost
  as a summation over the variables *(objective)*
- Collect costs, capacities, supplies, demands *(data)*

Send to a solver that computes solutions

- Available ready to run, without programming
- Handles very broad problem classes efficiently
  - Ex: Linear constraints and objective, continuous or integer variables
- Exploits provably optimal algorithms

Supply-Chain Optimization

Model-Based Formulation

Given

P set of products

N set of network locations

$A \subseteq N \times N$ set of links connecting locations

and

u_{ij} capacity of link from i to j, for each $(i, j) \in A$

s_{pj} supply/demand of product p at location j, for each $p \in P$, $j \in N$
> 0 implies supply, < 0 implies demand

c_{pij} cost per unit to ship product p on link (i, j),
for each $p \in P$, $(i, j) \in A$

Supply-Chain Optimization

Model-Based Formulation *(cont'd)*

Determine

X_{pij} amount of product p to be shipped from location i to location j, for each $p \in P$, $(i, j) \in A$

to minimize

$\sum_{p \in P} \sum_{(i,j) \in A} c_{pij} X_{pij}$

total cost of shipments

subject to

$\sum_{p \in P} X_{pij} \leq u_{ij}$, for all $(i, j) \in A$

on each link, total shipped must not exceed capacity

$\sum_{(i,j) \in A} X_{pij} + s_{pj} = \sum_{(j,i) \in A} X_{pji}$, for all $p \in P$, $j \in N$

at each location, shipments in plus
supply/demand must equal shipments out

Complications:
Supply Chain Optimization

Additional restrictions imposed by the user

- Cost has fixed and variable parts
  - Each link incurs a cost if it is *used* for shipping
- Shipments cannot be too small
- Not too many links can be used

Additional data for the problem

d_{ij} fixed cost for using the link from i to j, for each $(i, j) \in A$

m smallest total that may be shipped on any link used

n largest number of links that may be used

Complications

Method-Based *(cont'd)*

What has to be done?

- ❖ Revise or re-think the solution approach
- ❖ Update or re-implement the method

What are the challenges?

- ❖ In this example,
  - ∗ Shipments have become more interdependent
  - ∗ Good routes are harder to identify
  - ∗ Improvements are harder to find
- ❖ In general,
  - ∗ Even small changes to a problem can necessitate major changes to the method and its implementation
  - ∗ Each problem change requires more method development

. . . and problem changes are frequent!

Model-Based *(cont'd)*

What has to be done?

- Update the objective expression
- Formulate additional constraint equations
- Send back to the solver

What are the challenges?

- In this example,
  - New variables and expressions to represent fixed costs
  - New constraints to impose shipment and arc-use limits
- In general,
  - The formulation tends to get more complicated
  - A new solver type or solver options may be needed

. . . but it's easier to update formulations than methods
. . . and a few solver types handle many formulations

Model-Based Formulation *(revised)*

Determine

X_{pij} amount of commodity p to be shipped on link (i, j),
for each $p \in P$, $(i, j) \in A$

Y_{ij} 1 if any amount is shipped from location i to location j,
0 otherwise, for each $(i, j) \in A$

to minimize

$\sum_{p \in P} \sum_{(i,j) \in A} c_{pij} X_{pij} + \sum_{(i,j) \in A} d_{ij} Y_{ij}$

total varying plus fixed cost of shipments

Complications

Model-Based Formulation *(revised)*

Subject to

$\sum_{p \in P} X_{pij} \leq u_{ij} Y_{ij}$, for all $(i, j) \in A$

when the link from location i to location j is used,
total shipments must not exceed capacity, and Y_{ij} must be 1

$\sum_{(i,j) \in A} X_{pij} + s_{pj} = \sum_{(j,i) \in A} X_{pji}$, for all $p \in P, j \in N$

shipments in plus supply/demand must equal shipments out

$\sum_{p \in P} X_{pij} \geq m Y_{ij}$, for all $(i, j) \in A$

when the link from node i to node j is used,
total shipments from i to j must be at least m

$\sum_{(i,j) \in A} Y_{ij} \leq n$

At most n links can be used

Approaches to Model-Based Optimization

Translate between two forms of the problem

- Modeler's form
  - Symbolic description, easy for people to work with
- Solver's form
  - Explicit data structure, easy for solvers to compute with

Programming language approach

- Write a *computer program* to generate the solver's form

Modeling language approach

- Write the *model formulation*
  in a form that a computer can read and translate

Approaches to Modeling Languages

Algebraic modeling languages

- Designed for "algebraic" formulations
  as seen in our model-based examples
- Good fit to many applications and many solvers

Executable approach

- Write a *computer program* . . .
  - that resembles an optimization model
  - that can be executed to drive a solver

Declarative approach

- Write a *model description* . . .
  - in a language specialized for optimization
  - that can be translated to the solver's form

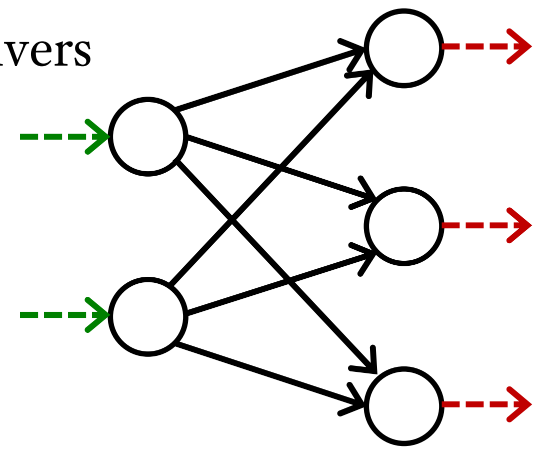
Example:
Supply Chain Optimization

Executable approach: gurobipy

- Based on the Python programming language
  - Designed to look like algebraic notation
- Generates problems for the Gurobi solver

Declarative approach: AMPL

- Based directly on algebraic notation
  - Designed specifically for optimization
- Generates problems for Gurobi and other solvers



Multi-Product Flow

Formulation: Data

Given

P set of products

N set of network nodes

$A \subseteq N \times N$ set of arcs connecting nodes

and

u_{ij} capacity of arc from i to j, for each $(i, j) \in A$

s_{pj} supply/demand of product p at node j, for each $p \in P$, $j \in N$
> 0 implies supply, < 0 implies demand

c_{pij} cost per unit to ship product p on arc (i, j),
for each $p \in P$, $(i, j) \in A$

Multi-Product Flow

Statements: Data

gurobipy

- Assign values to Python lists and dictionaries

```
products = ['Pencils', 'Pens']

nodes = ['Detroit', 'Denver',
 'Boston', 'New York', 'Seattle']

arcs, capacity = multidict({
 ('Detroit', 'Boston'):   100,
 ('Detroit', 'New York'):  80,
 ('Detroit', 'Seattle'):  120,
 ('Denver',  'Boston'):   120,
 ('Denver',  'New York'): 120,
 ('Denver',  'Seattle’):  120 })
```

- Provide data later in a separate file →

AMPL

- Define symbolic model sets and parameters

```
set PRODUCTS;
set NODES;

set ARCS within {NODES,NODES};
param capacity {ARCS} >= 0;
```

```
set PRODUCTS := Pencils Pens ;

set NODES := Detroit Denver
 Boston 'New York' Seattle ;

param: ARCS: capacity:
        Boston 'New York' Seattle :=
 Detroit    100       80       120
 Denver     120      120       120  ;
```

Multi-Product Flow

Formulation: Model

Determine

X_{pij} amount of commodity p to be shipped from node i to node j, for each $p \in P$, $(i,j) \in A$

to minimize

$\sum_{p \in P} \sum_{(i,j) \in A} c_{pij} X_{pij}$

total cost of shipping

subject to

$\sum_{p \in P} X_{pij} \leq u_{ij}$, for all $(i,j) \in A$

total shipped on each arc must not exceed capacity

$\sum_{(i,j) \in A} X_{pij} + s_{pj} = \sum_{(j,i) \in A} X_{pji}$, for all $p \in P$, $j \in N$

shipments in plus supply/demand must equal shipments out

Multi-Product Flow

Statements: Model

gurobipy

```
m = Model('netflow')

flow = m.addVars(products, arcs, obj=cost, name="flow")

m.addConstrs(
   (flow.sum('*',i,j) <= capacity[i,j] for i,j in arcs), "cap")

m.addConstrs(
   (flow.sum(p,'*',j) + inflow[p,j] == flow.sum(p,j,'*')
       for p in products for j in nodes), "node")
```

$$\sum_{(i,j)\in A} X_{pij} + s_{pj} = \sum_{(j,i)\in A} X_{pji}, \quad \text{for all } p \in P,\, j \in N$$

Multi-Product Flow

Statements: Model *(cont'd)*

AMPL

```
var Flow {PRODUCTS,ARCS} >= 0;

minimize TotalCost:
   sum {p in PRODUCTS, (i,j) in ARCS} cost[p,i,j] * Flow[p,i,j];

subject to Capacity {(i,j) in ARCS}:
   sum {p in PRODUCTS} Flow[p,i,j] <= capacity[i,j];

subject to Conservation {p in PRODUCTS, j in NODES}:
   sum {(i,j) in ARCS} Flow[p,i,j] + inflow[p,j] =
   sum {(j,i) in ARCS} Flow[p,j,i];
```

$$\sum_{(i,j)\in A} X_{pij} + s_{pj} = \sum_{(j,i)\in A} X_{pji}, \quad \text{for all } p \in P,\ j \in N$$

Multi-Product Flow

Solution

gurobipy

```
m.optimize()

if m.status == GRB.Status.OPTIMAL:
    solution = m.getAttr('x', flow)
        for p in products:
            print('\nOptimal flows for %s:’ % p)
            for i,j in arcs:
                if solution[p,i,j] > 0:
                    print('%s -> %s: %g' % (i, j, solution[p,i,j]))
```

```
Solved in 0 iterations and 0.00 seconds
Optimal objective  5.500000000e+03

Optimal flows for Pencils:
Detroit -> Boston: 50
Denver -> New York: 50
Denver -> Seattle: 10

Optimal flows for Pens: ...
```

Multi-Product Flow

Solution *(cont'd)*

AMPL

```
ampl: model netflow.mod;
ampl: data netflow.dat;

ampl: option solver gurobi;
ampl: solve;

Gurobi 9.0.3: optimal solution; objective 5500
2 simplex iterations

ampl: display Flow;

Flow [Pencils,*,*]
:        Boston 'New York' Seattle    :=
Denver      0        50        10
Detroit    50         0         0

 [Pens,*,*]
:        Boston 'New York' Seattle    :=
Denver     10         0        30
Detroit    30        30         0
;
```

Multi-Product Flow

Solution *(cont'd)*

AMPL

```
ampl: model netflow.mod;
ampl: data netflow.dat;

ampl: option solver cplex;
ampl: solve;

CPLEX 12.10.0.0: optimal solution; objective 5500
0 dual simplex iterations (0 in phase I)

ampl: display Flow;

Flow [Pencils,*,*]
:        Boston 'New York' Seattle    :=
Denver      0        50        10
Detroit    50         0         0

 [Pens,*,*]
:        Boston 'New York' Seattle    :=
Denver     10         0        30
Detroit    30        30         0
;
```

Executable

Concept

- Create an algebraic modeling language
  inside a general-purpose programming language
- Redefine operators like + and <=
  to return constraint objects rather than simple values

Advantages

- Complete application development in one language
- Direct access to advanced solver features

Disadvantages

- Programming languages are not designed for describing models
  - Constraint descriptions can be awkward
  - Special methods may be required for efficiency
- Modeling and programming bugs are hard to separate

Declarative

Concept

- Design a language for describing optimization models
- Connect to external applications via . . .
  - extensions for scripting and data transfer
  - APIs for programming languages

Disadvantages

- Adds a system between application and solver

Advantages

- Designed for building and using optimization models
  - Streamlines model building and processing
  - Promotes validation and maintenance of models
- Not specific to one programming language or solver

Integration with Applications

gurobipy

- Everything can be developed in Python
- Part of the Gurobi package
  - Free solver-independent alternatives (Pyomo, PuLP, Python-MIP)

AMPL

- Prototypes can be developed in AMPL
  - Modeling language extended with loops, tests, assignments
- Application programming interfaces (APIs)
  for integrating AMPL with popular programming languages
  - C++, C#, Java, MATLAB, Python, R

Integration with Solvers

gurobipy

- Works closely with the Gurobi solver:
  callbacks during optimization, fast re-solves after problem changes
- Supports Gurobi's extended expressions:
  min/max, and/or, if-then-else

AMPL

- Supports all popular solvers
- Extends to general nonlinear and logic expressions
  - Connects to nonlinear function libraries and user-defined functions
  - Automatically computes nonlinear function derivatives
  - Connects to global optimization and constraint programming solvers

Case Studies

Young's Plant Farm

- Packing shipments

Motion Robotics

- Designing aircraft

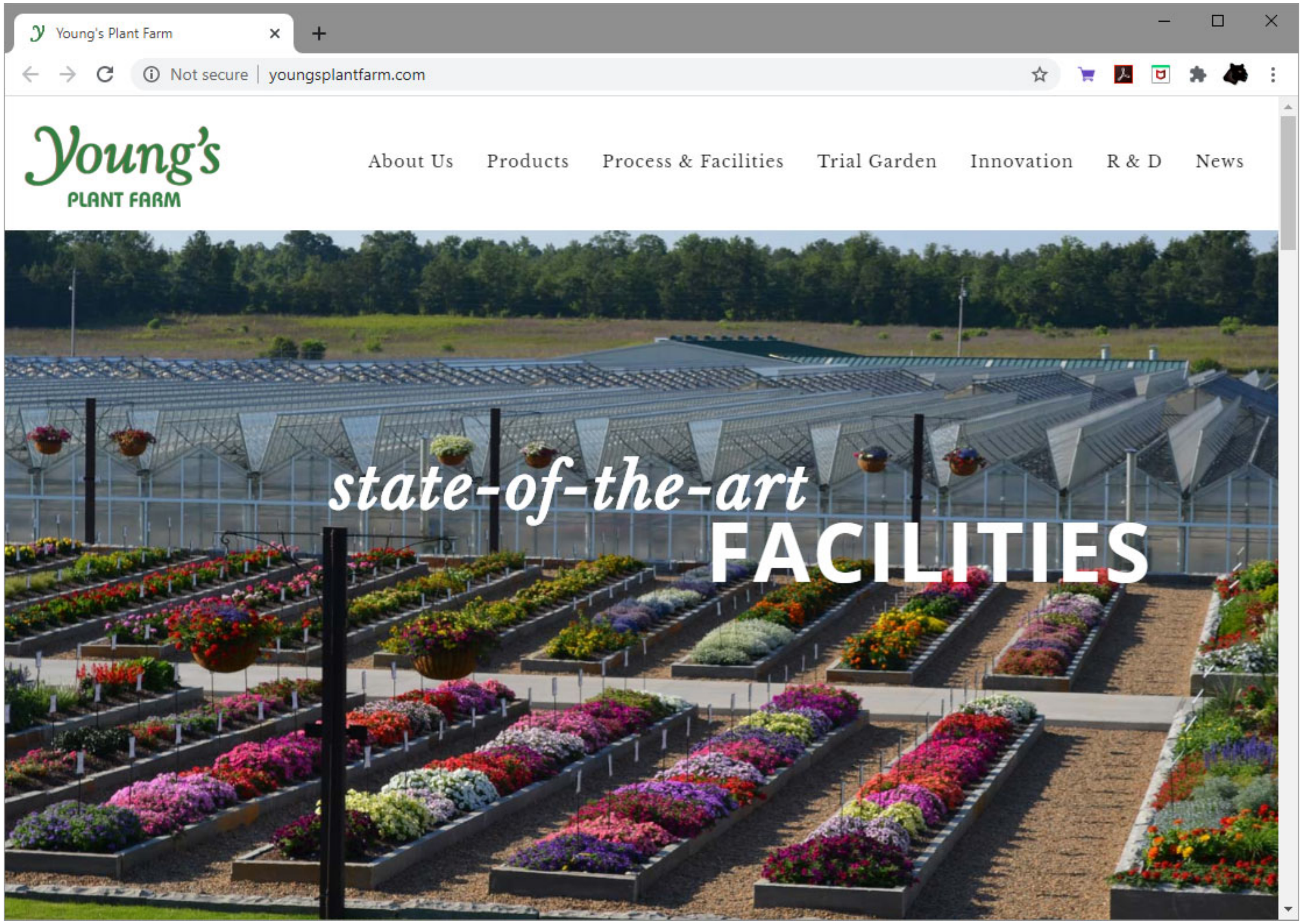
ABB

- Managing power grids

New York Education Department

- Assigning students to classes

Case: Young's Plant Farm
Packing Shipments



Packing

Situation

- ❖ Grows plants of many kinds and sizes
- ❖ Ships to retailers on their own trucks
  - ∗ Large customers include Walmart, Lowe's
- ❖ Plants are packed on special rolling racks
  - ∗ 3 feet wide, 4 feet long, 7 feet tall
  - ∗ 4 to 12 shelves

Goal

- ❖ Generate good packing plans for a day's orders
  - ❖ Don't use more racks than needed
- ❖ Finish in time to get the orders out

Evaluation

Approaches considered

- Spreadsheet “by hand”
- Algebraic modeling language + integer linear solver

Choice of AMPL

- Dramatically better solutions
- Numerous economies
  - Faster solutions using many fewer people
  - Faster loading of racks
  - Fewer trucks required
- Selection of solvers

Implementation

Development

- Original model built by Prof. Rafay Ishfaq of Auburn University
- Extended to handle larger orders by AMPL Optimization

Optimization

- Implemented using AMPL model, data, and scripts
- Minimum size: low 100s of thousands of variables & constraints
  Maximum size: 100 *million* variables & constraints
- Solve time: 10 to 45 minutes

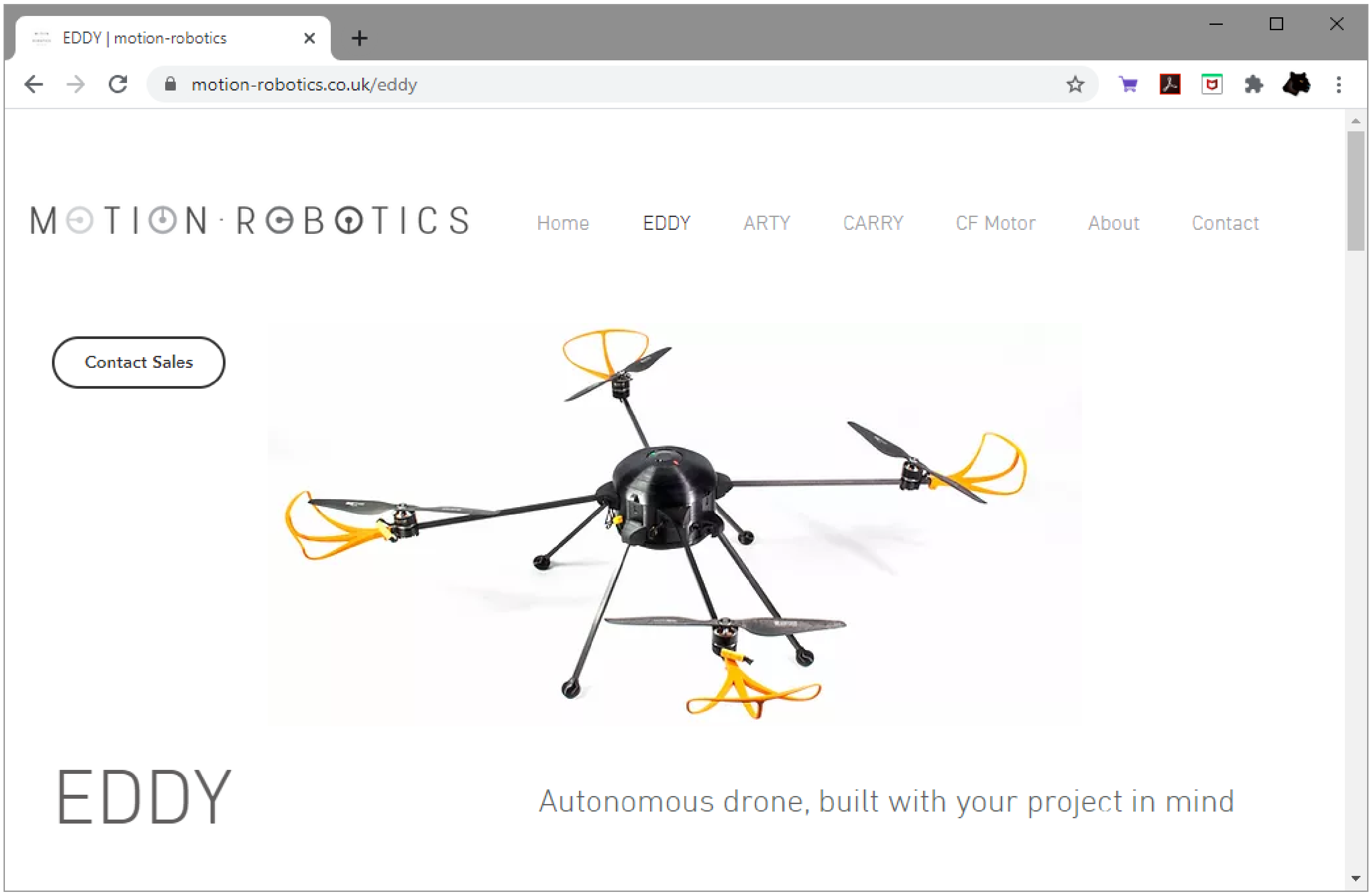
Deployment

- VBA-modified spreadsheet for data prep and result reporting
- One replenishment specialist uses the tool multiple times a day

. . . considering adaptations to new use cases

Case: Motion Robotics

Designing Aircraft



Design

Situation

- Develops and sells electric flying vehicles
  - Drones and related infrastructure (such as docking stations)
  - Electric motors using a highly efficient radial (vs. traditional axial) design
- Designs drones for specific applications

Goal

- Create new electric aircraft
- Evaluate over many possibilities
  - Design: propulsion, aerodynamics, structures
  - Architecture: rotor size, battery capacity, etc.
  - Flight path: initial & final position, velocity, etc.

Evaluation

Approaches considered

- Spreadsheet (not realistic)
- Simulation and machine learning
- Python-based design optimization system
- Algebraic modeling language + nonlinear solver

Choice of AMPL

- Formal optimization model
  - Generality to implement complex physics equations
  - Best results within a reasonable time frame
- Variety of solvers
- Ability to deploy via API
  - Import data
  - Run many optimizations in parallel

Implementation

Development

- ❖ Suggested by a professor at a local university
- ❖ Implemented by two analysts at Motion Robotics
- ❖ 1D prototypes built within months
- ❖ Expanded to 2D with many complicating factors

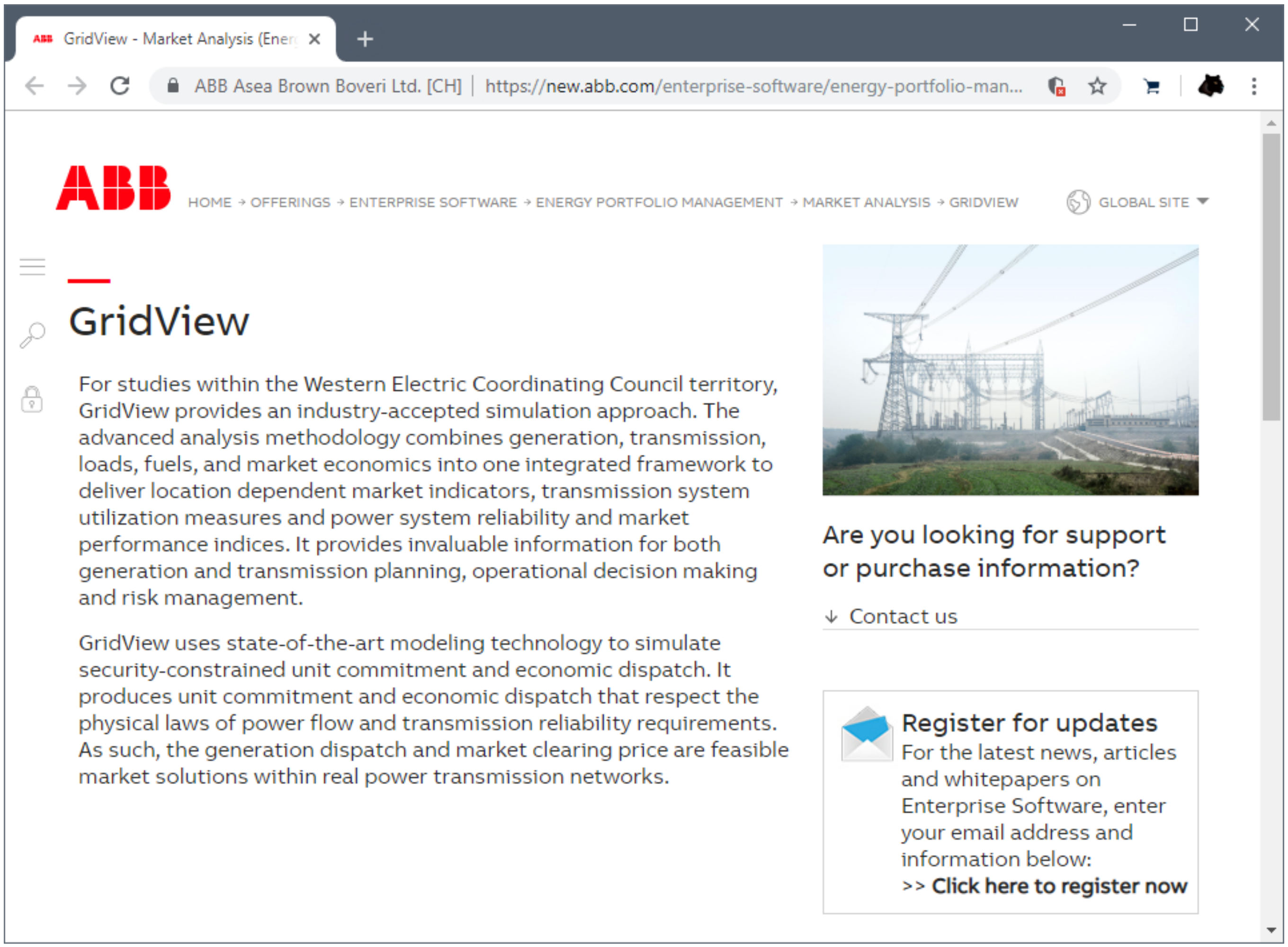
Optimization

- ❖ 7 minutes for a solve
  - ∗ Tested five nonlinear solvers, chose LOQO
- ❖ AMPL scripts for core application
- ❖ Python API for data & results

Deployment

- ❖ Parallel runs on 2,000 nodes in a cluster
- ❖ Connections to MATLAB, TensorFlow, etc.

Case: ABB
Managing Power Grids



Case: ABB
Power Grid Management

GridView

For studies within the Western Electric Coordinating Council territory, GridView provides an industry-accepted simulation approach. The advanced analysis methodology combines generation, transmission, loads, fuels, and market economics into one integrated framework to deliver location dependent market indicators, transmission system utilization measures and power system reliability and market performance indices. It provides invaluable information for both generation and transmission planning, operational decision making and risk management.

GridView uses state-of-the-art modeling technology to simulate security-constrained unit commitment and economic dispatch. It produces unit commitment and economic dispatch that respect the physical laws of power flow and transmission reliability requirements. As such, the generation dispatch and market clearing price are feasible market solutions within real power transmission networks.

Power Grid Management

Situation

- A power grid operator providing electrical service
- Two kinds of decisions
  - *Unit commitment:* When to turn power plants on and off
  - *Network flow:* How to transmit power over the grid to meet demand

Goal

- Simulate optimal decisions to support planning
  - Transmission network expansion
  - Plant addition and retirement
  - Integration of renewable energy sources

Evaluation

Approaches considered

- C++ for entire GridView system
- Modeling language for optimization, C++ for user interfaces

Choice of AMPL

- *Ease of modeling*
  - ABB can formulate complex and powerful models
  - Customers can understand the AMPL formulations
  - Customers can customize models for their particular situations
- *Ease of embedding*
  - AMPL has an API (application programming interface) for C++
  - ABB can easily build AMPL into the GridView product

Power

Implementation

Development

- Prototype at University of Tennessee, Knoxville
- Full AMPL implementation by three analysts at ABB

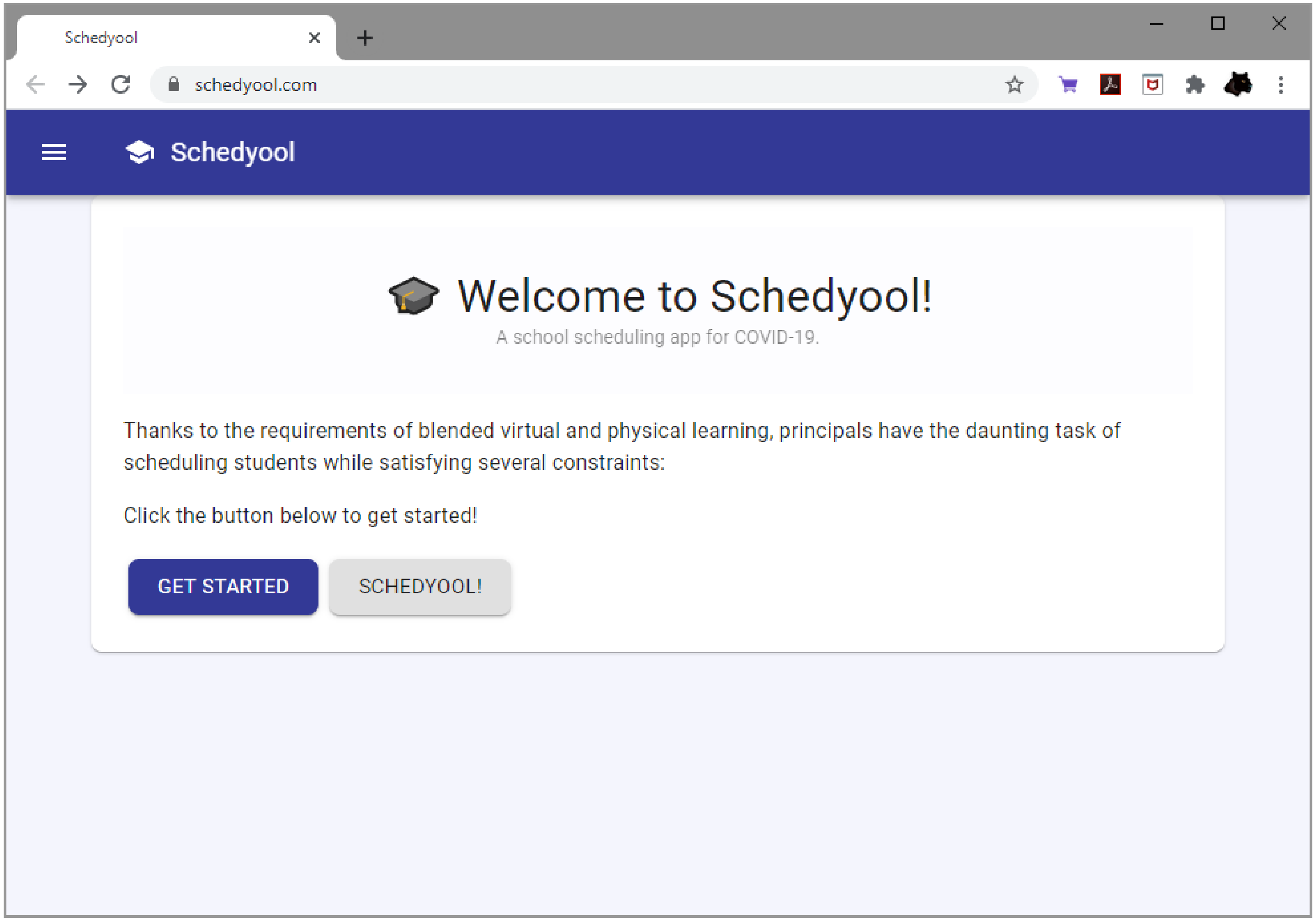
Optimization

- Mixed-integer linear solver
- Millions of variables
- Tens of thousands of integer variables
- 10 minutes to solve

Deployment

- 30+ customer companies
- Hundreds of customer-side users

Case: New York Education Department
Assigning Students to Classes



Assignment

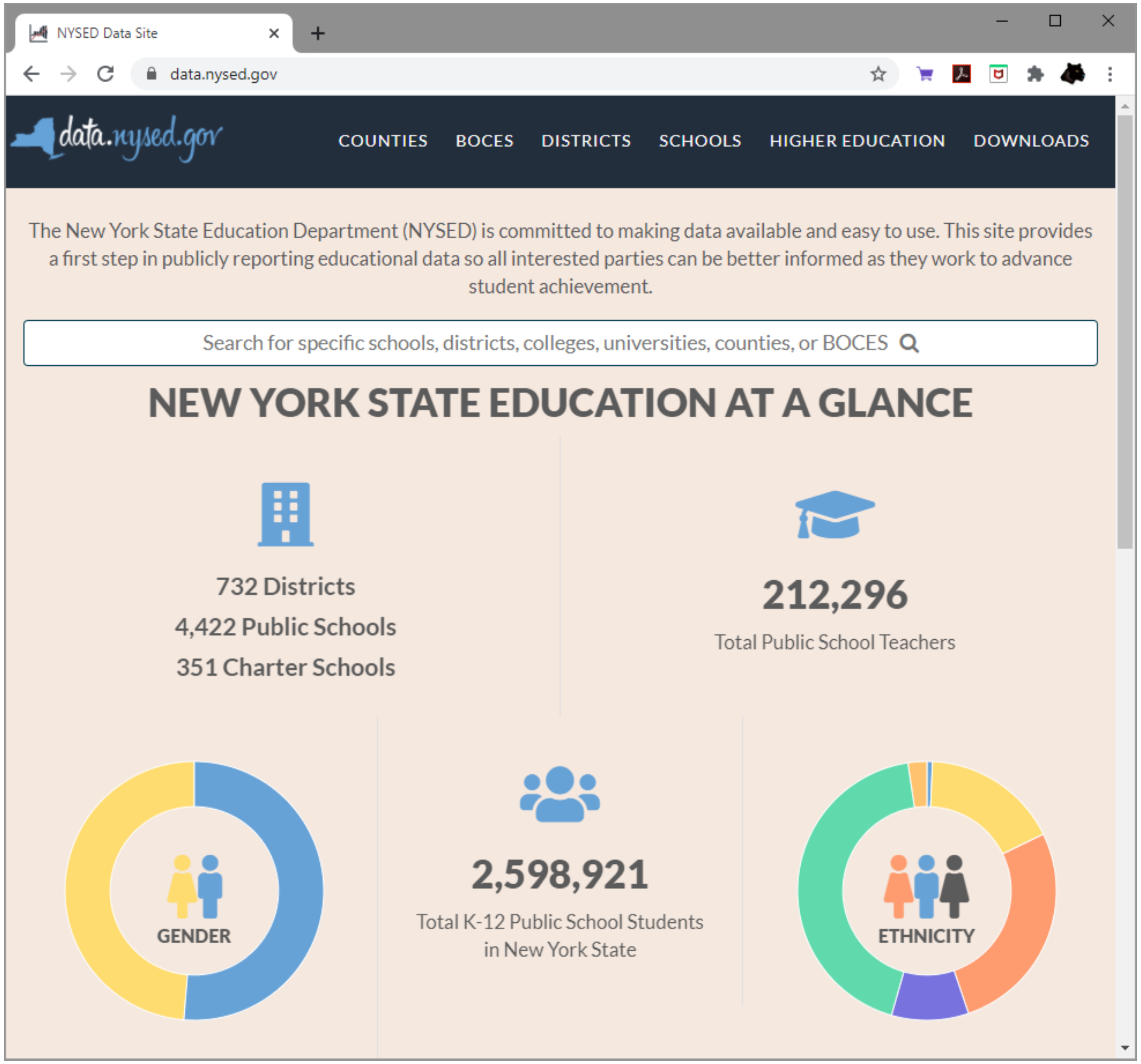
Situation

- Covid-19 closed New York State public schools in Spring 2020
- Now the schools want to reopen for Fall
- To limit the spread of the virus . . .
  - classrooms can hold only a limited number of students
  - state is offering "blended" instruction:
    "cohorts" of students can attend only 1-3 days each week

Goal

- Create a tool for making workable assignments
- Make the tool usable by school principals, on short notice
- Support many simultaneous users . . .

Assignment


NYSED Data Site
data.nysed.gov
data.nysed.gov
COUNTIES BOCES DISTRICTS SCHOOLS HIGHER EDUCATION DOWNLOADS
The New York State Education Department (NYSED) is committed to making data available and easy to use. This site provides a first step in publicly reporting educational data so all interested parties can be better informed as they work to advance student achievement.
Search for specific schools, districts, colleges, universities, counties, or BOCES
NEW YORK STATE EDUCATION AT A GLANCE
732 Districts
4,422 Public Schools
351 Charter Schools
212,296
Total Public School Teachers
GENDER
2,598,921
Total K-12 Public School Students in New York State
ETHNICITY

Evaluation

Approaches considered

- Every principal figures out how to make their own assignment
- Algebraic modeling language + integer linear solver

Choice of AMPL

- Fast prototyping and development
- Flexibility of solver choice
- Python API for deployment in easy-to-use web tool

Assignment

Implementation

Development

- Dr. Howard Karloff, Vice President, Goldman Sachs
  - model completed in a few days
- Caleb Ren, Harvard University student
- Filipe Brandão, AMPL Optimization

Optimization

- Tens of thousands of variables
- 1 to 4 minutes to solve

Deployment

- Easy-to-use web tool for data input and result reporting
  - click “submit” to run
- Python application parses data, runs AMPL and Gurobi
  - new AWS container spawned for each submission
  - application built in 2 weeks