Adding Optimization to Your Applications
Quickly and Reliably

1. A Guide to Model-Based Optimization

2. From Prototyping to Integration with AMPL

Robert Fourer

`4er@ampl.com`

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

INFORMS Business Analytics Conference
Technology Workshop, 3 April 2022

Outline

1. Model-based optimization

2. From prototyping to integration

- Building models: *AMPL's interactive environment*
- Developing optimization-based procedures: *AMPL scripts*
- Integrating into decision-making systems: *AMPL APIs*

3. Case studies

- *assignment:* Dropbox
- *packing:* Young's Plant Farm
- *deployment:* ABB / Hitachi Energy

Example: Roll Cutting

Motivation

- Fill orders for rolls of various widths
  - by cutting raw rolls of one (large) fixed width
  - using a variety of cutting patterns

Optimization model

- Decision variables
  - number of raw rolls to cut according to each pattern
- Objective
  - minimize number of raw rolls used
- Constraints
  - meet demands for each ordered width

Roll Cutting

Mathematical Formulation

Given

W	set of ordered widths
n	number of patterns considered

and

a_{ij}	occurrences of width i in pattern j, for each $i \in W$ and $j = 1, \ldots, n$
b_i	orders for width i, for each $i \in W$

Roll Cutting

Mathematical Formulation *(cont'd)*

Determine

X_j number of rolls to cut using pattern j,
for each $j = 1, \ldots, n$

to minimize

$\sum_{j=1}^{n} X_j$

total number of rolls cut

subject to

$\sum_{j=1}^{n} a_{ij} X_j \geq b_i$, for all $i \in W$

number of rolls cut of width i
must be at least the number ordered

Roll Cutting

AMPL Formulation

Symbolic model

```
set WIDTHS;
param orders {WIDTHS} > 0;

param nPAT integer >= 0;
param nbr {WIDTHS,1..nPAT} integer >= 0;

var Cut {1..nPAT} integer >= 0;

minimize Number:
   sum {j in 1..nPAT} Cut[j];

subj to Fulfill {i in WIDTHS}:
   sum {j in 1..nPAT} nbr[i,j] * Cut[j] >= orders[i];
```

$$\sum_{j=1}^{n} a_{ij} X_j \geq b_i, \text{ for all } i \in W$$

AMPL Formulation *(cont'd)*

Explicit data (independent of model)

```
param: WIDTHS: orders :=
         6.77    10
         7.56    40
        17.46    33
        18.76    10 ;

param nPAT := 9 ;

param nbr:  1  2  3  4  5  6  7  8  9 :=
     6.77   0  1  1  0  3  2  0  1  4
     7.56   1  0  2  1  1  4  6  5  2
    17.46   0  1  0  2  1  0  1  1  1
    18.76   3  2  2  1  1  1  0  0  0 ;
```

Command Environment

Model + data = problem instance to be solved

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

ampl: option solver cplex;

ampl: solve;

CPLEX 20.1.0.0: optimal integer solution; objective 20
3 MIP simplex iterations
0 branch-and-bound nodes

ampl: option omit_zero_rows 1;
ampl: option display_1col 0;

ampl: display Cut;

4 13   7 4   9 3
```

Command Language *(cont'd)*

Solver choice independent of model and data

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

ampl: option solver gurobi;

ampl: solve;

Gurobi 9.5.1: optimal solution; objective 20
3 simplex iterations
1 branch-and-cut nodes

ampl: option omit_zero_rows 1;
ampl: option display_1col 0;

ampl: display Cut;

4 13   7 4   9 3
```

Command Language *(cont'd)*

Results available for browsing

```
ampl: display {j in 1..nPAT} sum {i in WIDTHS} i * nbr[i,j];

1 63.84   3 59.41   5 64.09   7 62.82   9 59.66      # material used
2 61.75   4 61.24   6 62.54   8 62.03                # in each pattern

ampl: display {j in 1..nPAT, i in WIDTHS: Cut[j] > 0} nbr[i,j];

:       4   7   9   :=                               # patterns used
6.77    0   0   4
7.56    1   6   2
17.46   2   1   1
18.76   1   0   0

ampl: display Fulfill.slack;

 6.77  2                                             # overruns
 7.56  3                                             # of each width
17.46  0
18.76  3
```

Roll Cutting

Revision 1

Symbolic model

```
param roll_width > 0;

set WIDTHS;
param orders {WIDTHS} > 0;

param nPAT integer >= 0;
param nbr {WIDTHS,1..nPAT} integer >= 0;

var Cut {1..nPAT} integer >= 0;

minimize Number:
   sum {j in 1..nPAT} Cut[j];

minimize Waste:
   sum {j in 1..nPAT}
      Cut[j] * (roll_width - sum {i in WIDTHS} i * nbr[i,j]);

subj to Fulfill {i in WIDTHS}:

   sum {j in 1..nPAT} nbr[i,j] * Cut[j] >= orders[i];
```

Revision 1 *(cont'd)*

Explicit data

```
param roll_width := 64.5;

param: WIDTHS: orders :=
          6.77    10
          7.56    40
         17.46    33
         18.76    10 ;

param nPAT := 9 ;

param nbr:  1  2  3  4  5  6  7  8  9 :=
     6.77   0  1  1  0  3  2  0  1  4
     7.56   1  0  2  1  1  4  6  5  2
    17.46   0  1  0  2  1  0  1  1  1
    18.76   3  2  2  1  1  1  0  0  0 ;
```

Revision 1 *(cont'd)*

Solutions

```
ampl: model cutRev1.mod;
ampl: data cutRev1.dat;

ampl: objective Number; solve;

Gurobi 9.5.1: optimal solution; objective 20
3 simplex iterations
1 branch-and-cut nodes

ampl: display Number, Waste;
Number = 20
Waste = 63.62

ampl: objective Waste; solve;

Gurobi 9.5.1: optimal solution; objective 15.62
2 simplex iterations
1 branch-and-cut nodes

ampl: display Number, Waste;
Number = 35
Waste = 15.62
```

Roll Cutting

Revision 2

Symbolic model

```
param roll_width > 0;
param over_lim integer >= 0;

set WIDTHS;
param orders {WIDTHS} > 0;

param nPAT integer >= 0;
param nbr {WIDTHS,1..nPAT} integer >= 0;

var Cut {1..nPAT} integer >= 0;

...

subj to Fulfill {i in WIDTHS}:
   orders[i] <= sum {j in 1..nPAT} nbr[i,j] * Cut[j]
             <= orders[i] + over_lim;
```

Roll Cutting

Revision 2 *(cont'd)*

Explicit data

```
param roll_width := 64.5;
param over_lim := 8 ;

param: WIDTHS: orders :=
          6.77    10
          7.56    40
         17.46    33
         18.76    10 ;

param nPAT := 9 ;

param nbr:  1  2  3  4  5  6  7  8  9 :=
     6.77   0  1  1  0  3  2  0  1  4
     7.56   1  0  2  1  1  4  6  5  2
    17.46   0  1  0  2  1  0  1  1  1
    18.76   3  2  2  1  1  1  0  0  0 ;
```

Revision 2 *(cont'd)*

Solutions

```
ampl: model cutRev2.mod;
ampl: data cutRev2.dat;

ampl: objective Number; solve;

Gurobi 9.5.1: optimal solution; objective 20
7 simplex iterations
1 branch-and-cut nodes

ampl: display Number, Waste;
Number = 20
Waste = 62.04

ampl: objective Waste; solve;

Gurobi 9.5.1: optimal solution; objective 40.57
5 simplex iterations
1 branch-and-cut nodes

ampl: display Number, Waste;
Number = 24
Waste = 40.57
```

Scripting

Bring the programmer to the modeling language

Extend existing modeling language syntax . . .

- ❖ Algebraic expressions
- ❖ Set indexing expressions
- ❖ Interactive commands

. . . with programming concepts

- ❖ Loops of various kinds
- ❖ If-then and If-then-else conditionals
- ❖ Assignments

Examples

- ❖ Tradeoffs between *number cut* and *waste*
- ❖ Cutting via *pattern enumeration*
- ❖ Cutting via *pattern generation*

Tradeoffs Between Objectives

Minimize rolls cut

- Set large overrun limit

Minimize waste

- Reduce overrun limit 1 roll at a time
- If there is a change in number of rolls cut
  - record total waste (increasing)
  - record total rolls cut (decreasing)
- Stop when no further progress possible
  - problem becomes infeasible
  - total rolls cut falls to the minimum
- Report table of results

Scripting

Parametric Analysis *(cont'd)*

Script (setup and initial solve)

```
model cutRev2.mod;
data cutRev2.dat;

set OVER default {} ordered by reversed Integers;

param minNumber;
param minNumWaste;
param minWaste {OVER};
param minWasteNum {OVER};

param prev_number default Infinity;

option solver gurobi;
option solver_msg 0;


objective Number;
solve >Nul;

let minNumber := Number;
let minNumWaste := Waste;

objective Waste;
```

Scripting

Parametric Analysis *(cont'd)*

Script (looping and reporting)

```
for {k in over_lim .. 0 by -1} {
   let over_lim := k;

   solve >Nul;

   if solve_result = 'infeasible' then break;

   if Number < prev_number then {
      let OVER := OVER union {k};
      let minWaste[k] := Waste;
      let minWasteNum[k] := Number;
      let prev_number := Number;
   }

   if Number = minNumber then break;

}

printf 'Min%3d rolls with waste%6.2f\n\n', minNumber, minNumWaste;
printf ' Over  Waste  Number\n';
printf {k in OVER}: '%4d%8.2f%6d\n', k, minWaste[k], minWasteNum[k];
```

Parametric Analysis *(cont'd)*

Script run

```
ampl: include cutWASTE.run

Min 20 rolls with waste 62.04

 Over  Waste  Number
  25   40.57    24
  19   43.01    23
  13   45.45    22
   7   47.89    21
   5   54.76    20

ampl:
```

Cutting *via* Pattern Enumeration

Build the pattern list, then solve

- Read general model
- Read data: demands, raw width
- *Compute data: all usable patterns*
- Solve problem instance

Scripting

Pattern Enumeration

Model

```
param roll_width > 0;

set WIDTHS ordered by reversed Reals;
param orders {WIDTHS} > 0;

param maxPAT integer >= 0;
param nPAT integer >= 0, <= maxPAT;

param nbr {WIDTHS,1..maxPAT} integer >= 0;

var Cut {1..nPAT} integer >= 0;

minimize Number:

   sum {j in 1..nPAT} Cut[j];

subj to Fulfill {i in WIDTHS}:

   sum {j in 1..nPAT} nbr[i,j] * Cut[j] >= orders[i];
```

Pattern Enumeration

Data

```
param roll_width := 64.50 ;

param: WIDTHS: orders :=

         6.77    10
         7.56    40
        17.46    33
        18.76    10 ;
```

Scripting

Pattern Enumeration

Script (initialize)

```
model cutPAT.mod;

param dsetname symbolic;
printf "\nEnter dataset name:\n";
read dsetname <-;

data (dsetname & ".dat");


model;
param curr_sum >= -1e-10;
param curr_width > 0;
param pattern {WIDTHS} integer >= 0;

let maxPAT := 100000000;

let nPAT := 0;
let curr_sum := 0;
let curr_width := first(WIDTHS);
let {w in WIDTHS} pattern[w] := 0;
```

Pattern Enumeration

Script (loop)

```
repeat {
   if curr_sum + curr_width <= roll_width then {
      let pattern[curr_width] := floor((roll_width-curr_sum)/curr_width);
      let curr_sum := curr_sum + pattern[curr_width] * curr_width;
      }
   if curr_width != last(WIDTHS) then
      let curr_width := next(curr_width,WIDTHS);
   else {
      let nPAT := nPAT + 1;
      let {w in WIDTHS} nbr[w,nPAT] := pattern[w];
      let curr_sum := curr_sum - pattern[last(WIDTHS)] * last(WIDTHS);
      let pattern[last(WIDTHS)] := 0;
      let curr_width := min {w in WIDTHS: pattern[w] > 0} w;
      if curr_width < Infinity then {
         let curr_sum := curr_sum - curr_width;
         let pattern[curr_width] := pattern[curr_width] - 1;
         let curr_width := next(curr_width,WIDTHS);
         }
      else break;
      }
   }
```

Scripting

Pattern Enumeration

Script (solve, report)

```
printf "\nAT LEAST %d ROLLS REQUIRED\n\n",
   ceil((sum {i in WIDTHS} i * orders[i]) / roll_width);

option solver gurobi;

solve;

printf "\n%5i patterns, %3i rolls", nPAT, sum {j in 1..nPAT} Cut[j];
printf "\n\n  Cut   ";
printf {j in 1..nPAT: Cut[j] > 0}: "%3i", Cut[j];
printf "\n\n";

for {i in WIDTHS} {
   printf "%7.2f ", i;
   printf {j in 1..nPAT: Cut[j] > 0}: "%3i", nbr[i,j];
   printf "\n";
   }
```

Scripting

Pattern Enumeration

Results

```
ampl: include cutPatEnum.run

AT LEAST 18 ROLLS REQUIRED

Gurobi 9.5.1: optimal solution; objective 18
4 simplex iterations
1 branch-and-cut nodes

43 patterns, 18 rolls

  Cut      2  1  1  4 10

  18.76    3  2  2  0  0
  17.46    0  1  0  3  2
   7.56    1  1  3  1  3
   6.77    0  0  0  0  1
```

Scripting

Pattern Enumeration

Data 2

```
param roll_width := 349 ;

param: WIDTHS: orders :=
       28.75     7
       33.75    23
       34.75    23
       37.75    31
       38.75    10
       39.75    39
       40.75    58
       41.75    47
       42.25    19
       44.75    13
       45.75    26 ;
```

Scripting

Pattern Enumeration

Results 2

```
ampl: include cutPatEnum.run

AT LEAST 34 ROLLS REQUIRED

Gurobi 9.5.1: optimal solution; objective 34
126 simplex iterations
1 branch-and-cut nodes

54508 patterns, 34 rolls

  Cut     8  2  1  1  1  1  4  5  1  2  2  5  1

  45.75   3  1  0  0  0  0  0  0  0  0  0  0  0
  44.75   1  3  0  0  0  0  0  0  0  0  0  0  0
  42.25   0  4  6  4  1  0  0  0  0  0  0  0  0
  41.75   4  0  0  0  1  6  2  0  0  0  0  0  0
  40.75   0  0  0  2  0  1  1  6  5  5  3  0  0
  39.75   0  0  0  0  0  0  2  0  1  0  4  4  2
  38.75   0  0  0  1  0  0  0  0  0  2  0  1  0
  37.75   0  0  1  0  0  0  2  0  1  0  0  4  1
  34.75   0  0  0  0  0  0  2  3  0  0  0  0  0
  33.75   0  0  0  0  7  0  0  0  2  2  2  0  6
  28.75   0  0  2  2  1  2  0  0  0  0  0  0  1
```

Scripting

Pattern Enumeration

Data 3

```
param roll_width := 172 ;

param: WIDTHS: orders :=
      25.000     5
      24.750    73
      18.000    14
      17.500     4
      15.500    23
      15.375     5
      13.875    29
      12.500    87
      12.250     9
      12.000    31
      10.250     6
      10.125    14
      10.000    43
       8.750    15
       8.500    21
       7.750     5 ;
```

Scripting

Pattern Enumeration

Results 3 (using 1% of generated patterns)

```
ampl: include cutPatEnum.run

AT LEAST 33 ROLLS REQUIRED

Gurobi 9.5.1: optimal solution; objective 33
493 simplex iterations
1 branch-and-cut nodes

273380 patterns, 33 rolls

  Cut      5  5  5  1  1  1  4  6  2  1  1  1

  25.00    1  0  0  0  0  0  0  0  0  0  0  0
  24.75    1  5  3  2  2  2  2  2  1  0  0  0
  18.00    0  1  0  2  2  0  0  0  1  3  0  0
  17.50    0  0  0  0  0  4  0  0  0  0  0  0
  .......
  10.12    0  0  2  0  0  0  0  0  0  2  2  0
  10.00    0  0  0  0  2  0  0  6  0  4  1  0
   8.75    1  0  0  1  0  0  2  0  1  1  0  2
   8.50    2  0  0  0  5  0  0  0  3  0  0  0
   7.75    0  1  0  0  0  0  0  0  0  0  0  0
```

Scripting in practice . . .

Large and complex scripts

- Multiple files
- Hundreds of statements
- Millions of statements executed

Coordination with enterprise systems

- Your system
  - writes data files
  - invokes `ampl optapp.run`
- AMPL's script
  - reads the data files
  - processes data, generates problems, invokes solvers
  - writes result files
- Your system
  - reads the result files

Scripting

Limitations

Scripts can be slow

- ❖ Interpreted, not compiled
- ❖ Very general set & data structures

Script programming constructs are limited

- ❖ Based on a declarative language
- ❖ Not object-oriented

Scripts are stand-alone

- ❖ Run in a separate AMPL environment
- ❖ Challenging to package, integrate, and deploy

APIs (application programming interfaces)

Bring the modeling language to the programmer

- Data and result management in a general-purpose programming language
- Modeling and solving through calls to AMPL

Available in all AMPL distributions

- ***Python*** 2.7, 3.*x*
  - pip install amplpy
- ***C++, C#, MATLAB, Java***
  - Download from https://ampl.com/products/api/
- ***R***
  - install.packages("Rcpp", type="source")
  - install.packages(
    "https://ampl.com/dl/API/rAMPL.tar.gz", repos=NULL)

Cutting *Revisited*

Hybrid approach

- Control & pattern enumeration from a programming language
- Model definition & modeling commands in AMPL

Key to examples: Python and R

- AMPL entities
- AMPL API Python/R objects
- AMPL API Python/R methods
- Python/R functions etc.

Pattern Enumeration in Python

Load & generate data, set up AMPL model

```
def cuttingEnum(dataset):
  from amplpy import AMPL

  # Read orders, roll_width, overrun

  exec(open(dataset+'.py').read(), globals())

  # Enumerate patterns

  widths = list(sorted(orders.keys(), reverse=True))
  patmat = patternEnum(roll_width, widths)

  # Set up model

  ampl = AMPL()
  ampl.option['ampl_include'] = 'models'
  ampl.read('cut.mod')
```

Pattern Enumeration in Python

Send data to AMPL

```
# Send scalar values

ampl.param['nPatterns'] = len(patmat)
ampl.param['overrun'] = overrun
ampl.param['rawWidth'] = roll_width

# Send order vector

ampl.set['WIDTHS'] = widths
ampl.param['order'] = orders

# Send pattern matrix

ampl.param['rolls'] = {
   (widths[i], 1+p): patmat[p][i]
   for i in range(len(widths))
   for p in range(len(patmat))
}
```

AMPL API

Pattern Enumeration in Python

Solve and get results

```
# Solve

ampl.option['solver'] = 'gurobi'
ampl.solve()

# Retrieve solution

CuttingPlan = ampl.var['Cut'].getValues()
cutvec = list(CuttingPlan.getColumn('Cut.val'))
```

Pattern Enumeration in Python

Display solution

```
# Prepare solution data

summary = {
    'Data': dataset,
    'Obj': int(ampl.obj['TotalRawRolls'].value()),
    'Waste': ampl.getValue(
                 'sum {p in PATTERNS} Cut[p] * \
                    (rawWidth - sum {w in WIDTHS} w*rolls[w,p])'
             )
}

solution = [
    (patmat[p], cutvec[p])
    for p in range(len(patmat))
    if cutvec[p] > 0
]


# Create plot of solution

cuttingPlot(roll_width, widths, summary, solution)
```

Pattern Enumeration in Python

Enumeration routine

```
def patternEnum(roll_width, widths, prefix=[]):
  from math import floor

  max_rep = int(floor(roll_width/widths[0]))

  if len(widths) == 1:
      patmat = [prefix+[max_rep]]

  else:
     patmat = []
     for n in reversed(range(max_rep+1)):
        patmat += patternEnum(roll_width-n*widths[0], widths[1:], prefix+[n])

  return patmat
```

AMPL API

Pattern Enumeration in Python

Plotting routine

```
def cuttingPlot(roll_width, widths, summ, solution):
  import numpy as np
  import matplotlib.pyplot as plt

  ind = np.arange(len(solution))
  acc = [0]*len(solution)

  colorlist = ['red','lightblue','orange','lightgreen',
               'brown','fuchsia','silver','goldenrod']
```

AMPL API

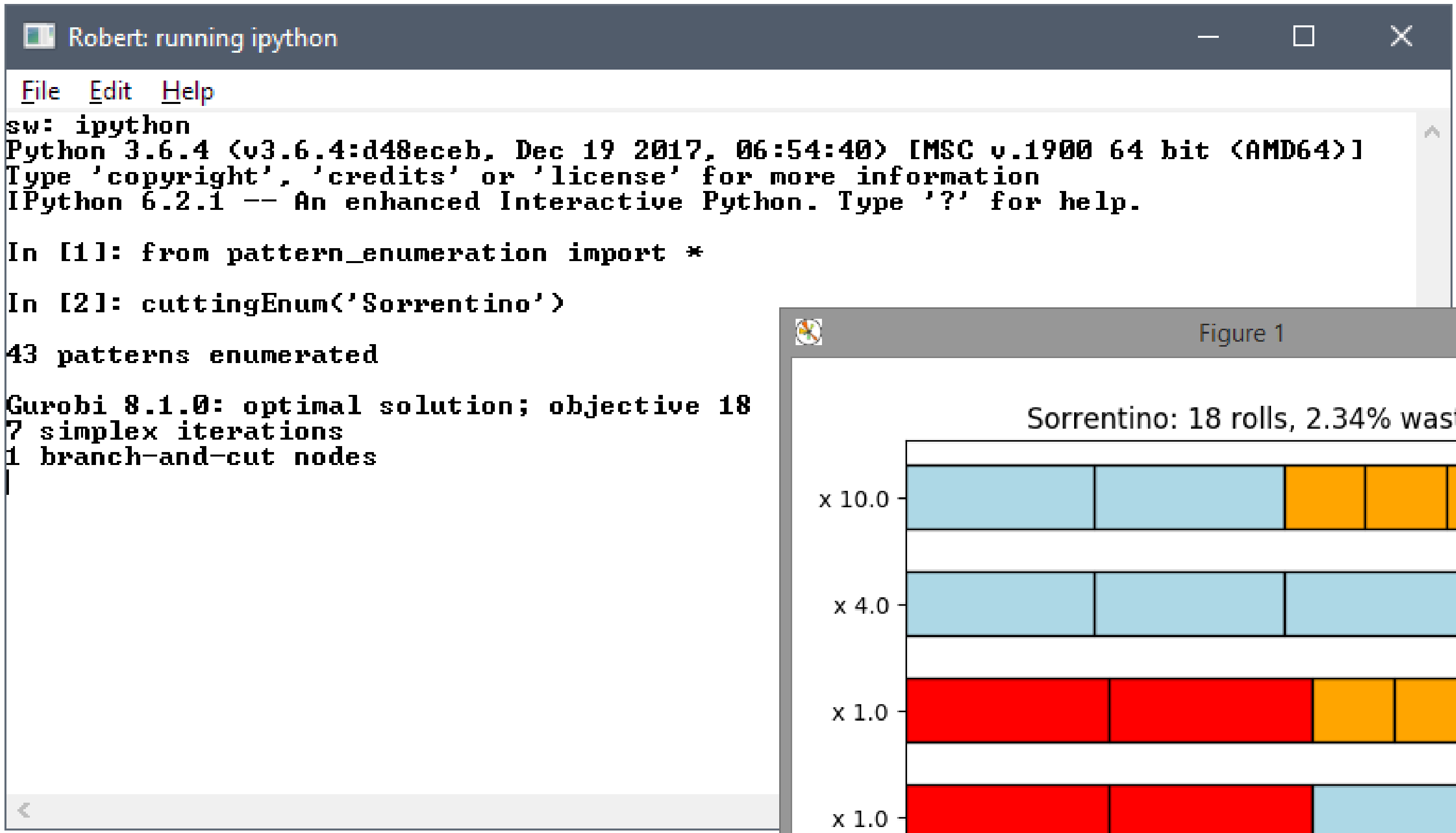
Pattern Enumeration in Python

Plotting routine (cont'd)

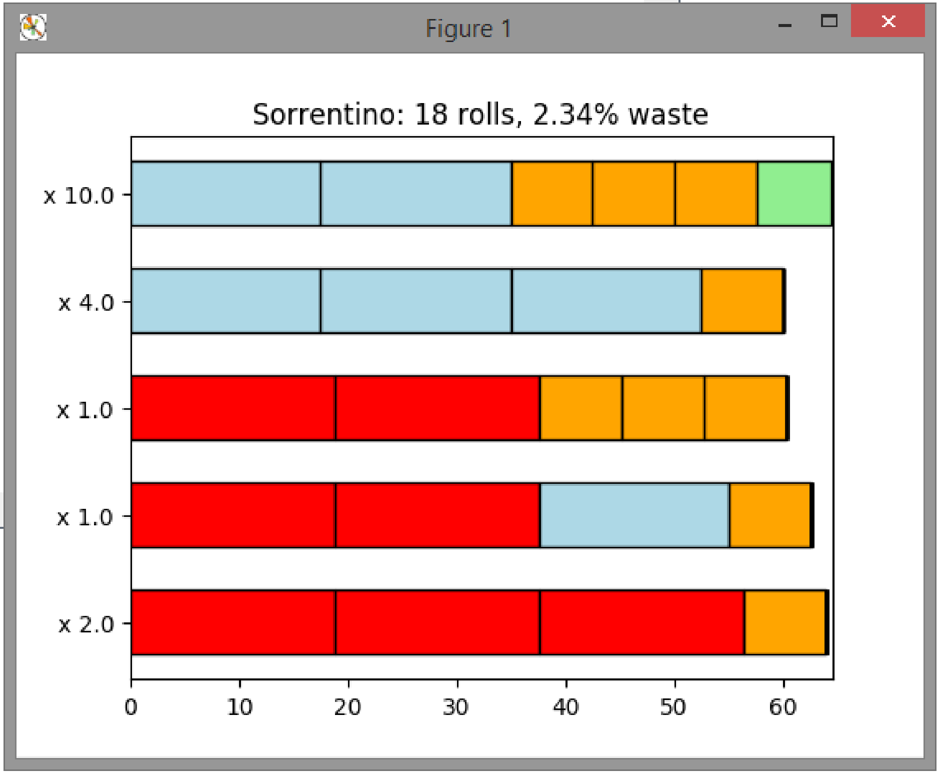
```
for p, (patt, rep) in enumerate(solution):
   for i in range(len(widths)):
      for j in range(patt[i]):
         vec = [0]*len(solution)
         vec[p] = widths[i]
         plt.barh(ind, vec, 0.6, acc,
                    color=colorlist[i%len(colorlist)], edgecolor='black')
         acc[p] += widths[i]

plt.title(summ['Data'] + ": " +
   str(summ['Obj']) + " rolls" + ", " +
   str(round(100*summ['Waste']/(roll_width*summ['Obj']),2)) + "% waste"
   )

plt.xlim(0, roll_width)
plt.xticks(np.arange(0, roll_width, 10))
plt.yticks(ind, tuple("x {:}".format(rep) for patt, rep in solution))

plt.show()
```

AMPL API

Pattern Enumeration in Python



```
sw: ipython
Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.2.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from pattern_enumeration import *

In [2]: cuttingEnum('Sorrentino')

43 patterns enumerated

Gurobi 8.1.0: optimal solution; objective 18
7 simplex iterations
1 branch-and-cut nodes
```



APIs in practice . . .

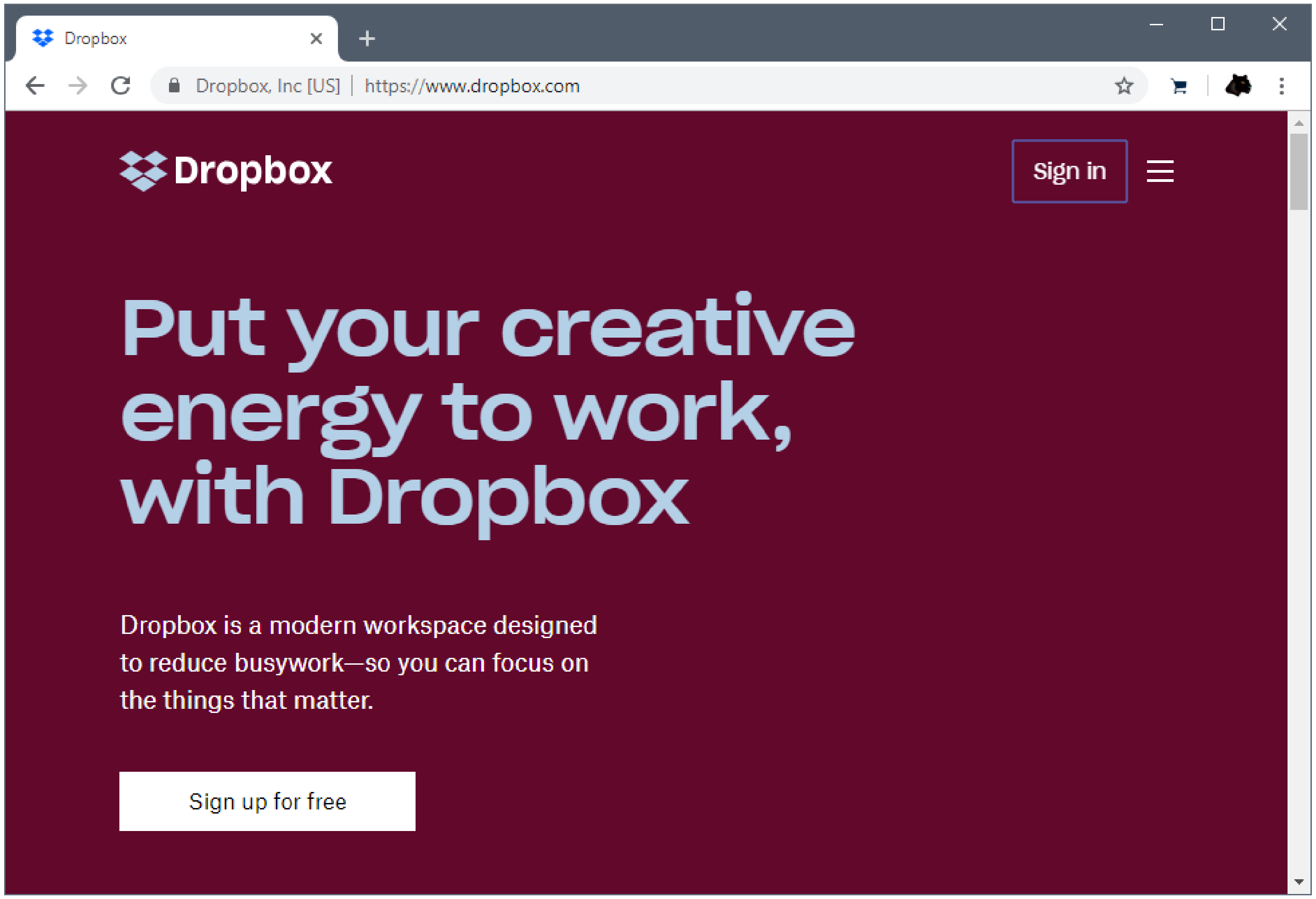
Application system in chosen programming language

- Extract requirements from enterprise database
- Generate good patterns to consider
- Feed results to visualization and deployment systems

Key role for modeling in AMPL

- Prototype and refine a model
- Evolve and maintain the model reliably
- Manage the interface to chosen solvers

Case: Dropbox
Sales Representative Assignment



Put your creative energy to work, with Dropbox

Dropbox is a modern workspace designed to reduce busywork—so you can focus on the things that matter.

Application

Setting

- ❖ Cloud storage provider
- ❖ Over 500 million users upload 1.2 billion files every day
- ❖ Tens of thousands of large business customer accounts
- ❖ Hundreds of sales representatives worldwide
  - ∗ enough to cover most but not all accounts

Goal

- ❖ Assign accounts to representatives
  - ∗ Assign each representative a similar number and quality of accounts
  - ∗ Give priority to assigning higher quality accounts

Sales Rep Assignment

Evaluation

Approaches considered

- ❖ Manual system
- ❖ Spreadsheet-based solvers
- ❖ Automated system using model-based optimization

Choice of AMPL

- ❖ Ease of use
- ❖ Speed
- ❖ Reliability
- ❖ Ability to handle large problems

Formulation *(data and variables)*

Data

- Quality *score* for each customer account
  - ∗ predicted revenue increase if contacted by a representative
- Location of each representative

Decision variables

- For each account i and representative j,
  $X_{ij} = 1$ if account i is assigned to representative j
  $X_{ij} = 0$ otherwise

Sales Rep Assignment

Formulation *(objective and constraints)*

Objective

- ❖ Maximize total score of all assigned accounts

Constraints

- ❖ At most 15% variance between representatives in . . .
  - ∗ number of accounts assigned
  - ∗ quality of accounts assigned
- ❖ Assigned accounts must be near the representative's location
- ❖ All subaccounts of a business must have the same representative

Sales Rep Assignment

Implementation

Development

- ❖ Implementation by 3 analysts at Dropbox

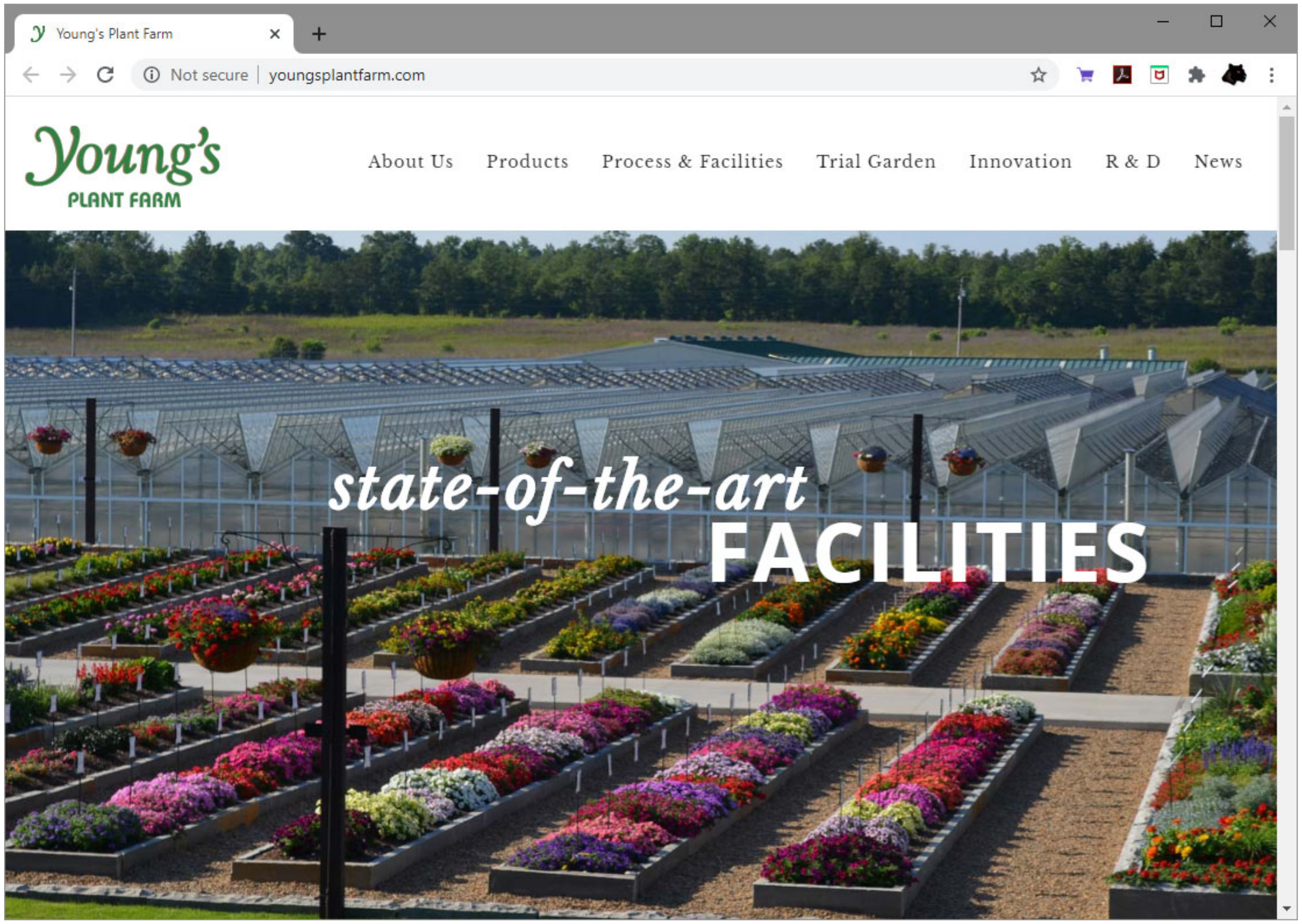
Optimization

- ❖ Mixed-integer linear solver
- ❖ 10,000 zero-one variables
- ❖ 3-6 hours to solve for largest region

Deployment

- ❖ 5-10 sales leaders are direct users
- ❖ AMPL is embedded in Dropbox’s systems
  - ∗ Customer data is extracted from *Salesforce*
  - ∗ Customer scores are computed using the *scikit-learn* Python toolbox
  - ∗ An AMPL script reads the file of score data
  - ∗ Results from optimization are written to an *Excel* spreadsheet

Case: Young's Plant Farm
Packing and Shipping



Application

Setting

- ❖ 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

Packing

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

Formulation

Data

- Set of stores
- Numbers of each plant ordered by each store
- Number of packing patterns: Ways that a rack may be packed
  - Up to several million, generated each day
- Number of each plant in each packing pattern
  - Up to six different plants per rack

Variables: all 0 or 1

- Whether a pattern is used at all
- Whether a store uses a certain pattern
- Whether a plant is shipped to a certain store using a certain pattern

Formulation

Objective: Minimize

- Racks used, plus penalties . . .
  - for not using all space in a pattern
  - for using all space in a tightly packed pattern

Constraints

- For each type of plant,
  the number of plants shipped to a store
  must equal the number of plants ordered by the store
- If a pattern is not used at all,
  then it cannot be used for any store
- If a pattern is not used for a particular store,
  then it cannot be used to send the store any of the plants it contains

Packing

Implementation

Development

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

Optimization

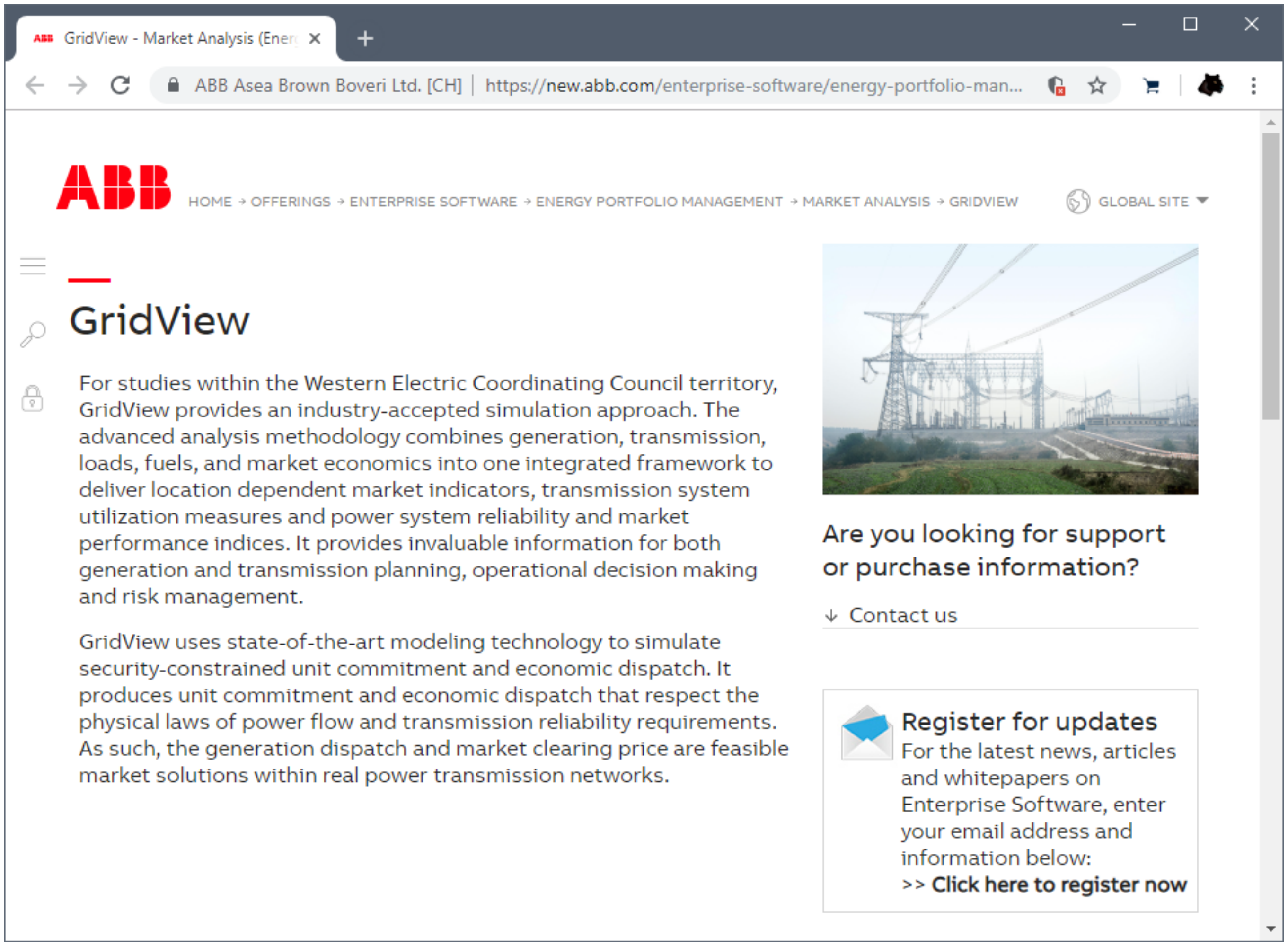
- ❖ Implemented using AMPL model, data, and scripts
- ❖ Minimum size: low 100s of thousands of variables & constraints
  Maximum size: 100 *million* variables & constraints
- ❖ Solve time: 10 to 45 minutes

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: ABB
Managing Power Grids


GridView - Market Analysis (Ener
ABB Asea Brown Boveri Ltd. [CH] | https://new.abb.com/enterprise-software/energy-portfolio-man...
HOME → OFFERINGS → ENTERPRISE SOFTWARE → ENERGY PORTFOLIO MANAGEMENT → MARKET ANALYSIS → GRIDVIEW
GLOBAL SITE

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.

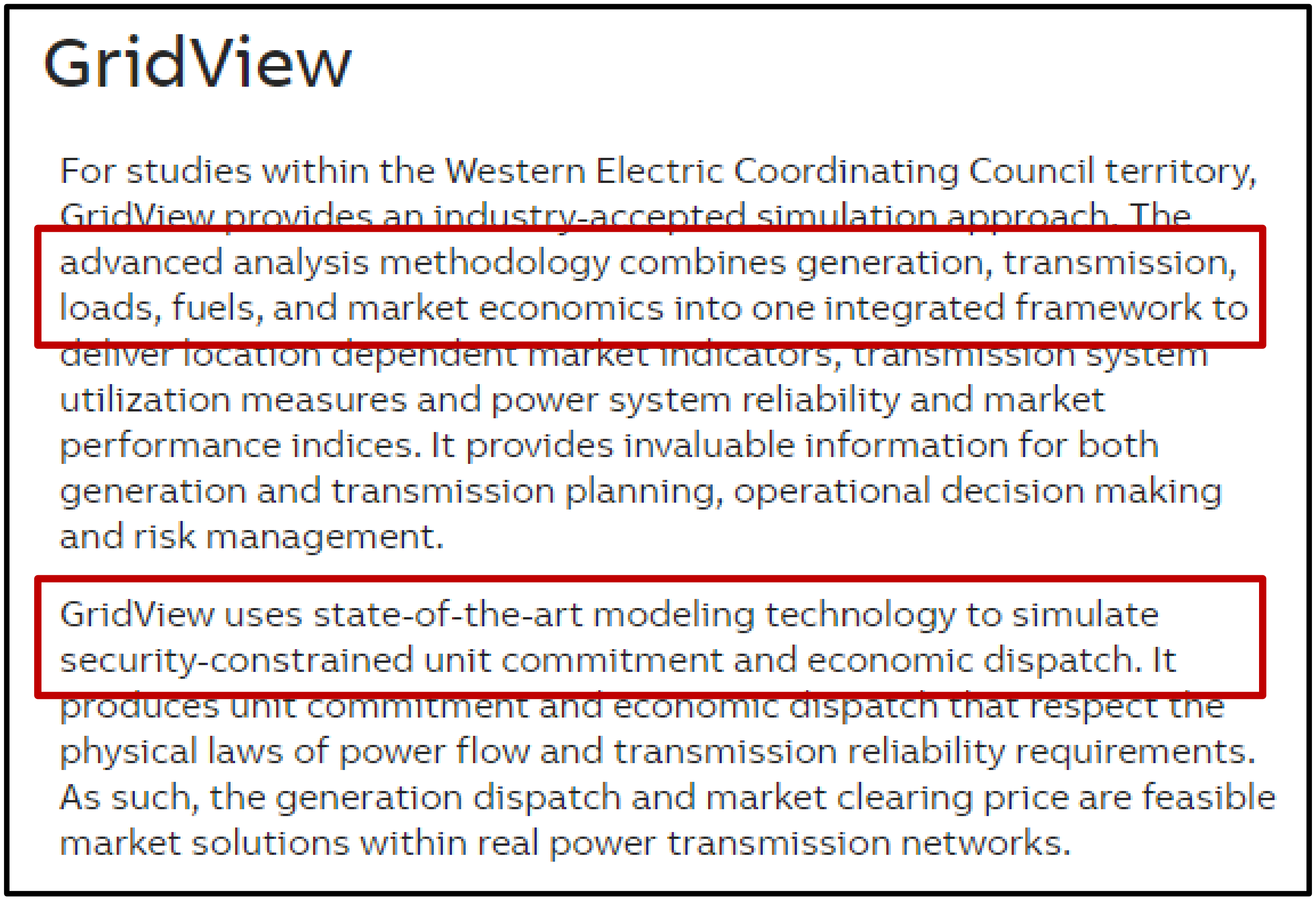
GridView uses state-of-the-art modeling technology to simulate security-constrained unit commitment and economic dispatch. It produces unit commitment and economic dispatch that respect the physical laws of power flow and transmission reliability requirements. As such, the generation dispatch and market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

Are you looking for support or purchase information?

↓ Contact us

Register for updates
For the latest news, articles and whitepapers on Enterprise Software, enter your email address and information below:
>> **Click here to register now**

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

Application

Setting

- ❖ Power grid operators provide 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

Power Grid

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 specialize models to their particular situations
- ❖ *Ease of embedding*
  - ∗ Optimization can be built into the GridView product using AMPL's C++ API

Power

Formulation *(data)*

Production data

- Power generation units
  - Location
  - Fuel, design, age, capacity
  - Ramp-up and ramp-down times
- Renewable energy sources

Transmission network data

- Nodes: units, sources, substations, customers
  - Supply at plants and other sources
  - Demand at customers
- Arcs: power lines
  - Transmission capacities

Cost data

Power

Formulation *(variables)*

Decision variables

- For each unit, in each time period
  - On or off (discrete)
  - Level of output (continuous)
- For each *critical path* through the grid, in each time period
  - Capacity

Power

Formulation *(model)*

Objectives

- ❖ For short-term operation management
  - ∗ Minimize total operating costs
- ❖ For long-term investment planning
  - ∗ Minimize total operating and investment costs

Constraints

- ❖ Balance of supply and demand
- ❖ Capacity restriction on power lines
- ❖ Ramp-up and ramp-down times
- ❖ Contingencies for generation and transmission

Power Grid

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: Hitachi Energy
Managing Power Grids

Simulate security-constrained unit commitment and economic dispatch in large-scale transmission networks

Simulate security-constrained unit commitment and economic dispatch in large-scale transmission networks

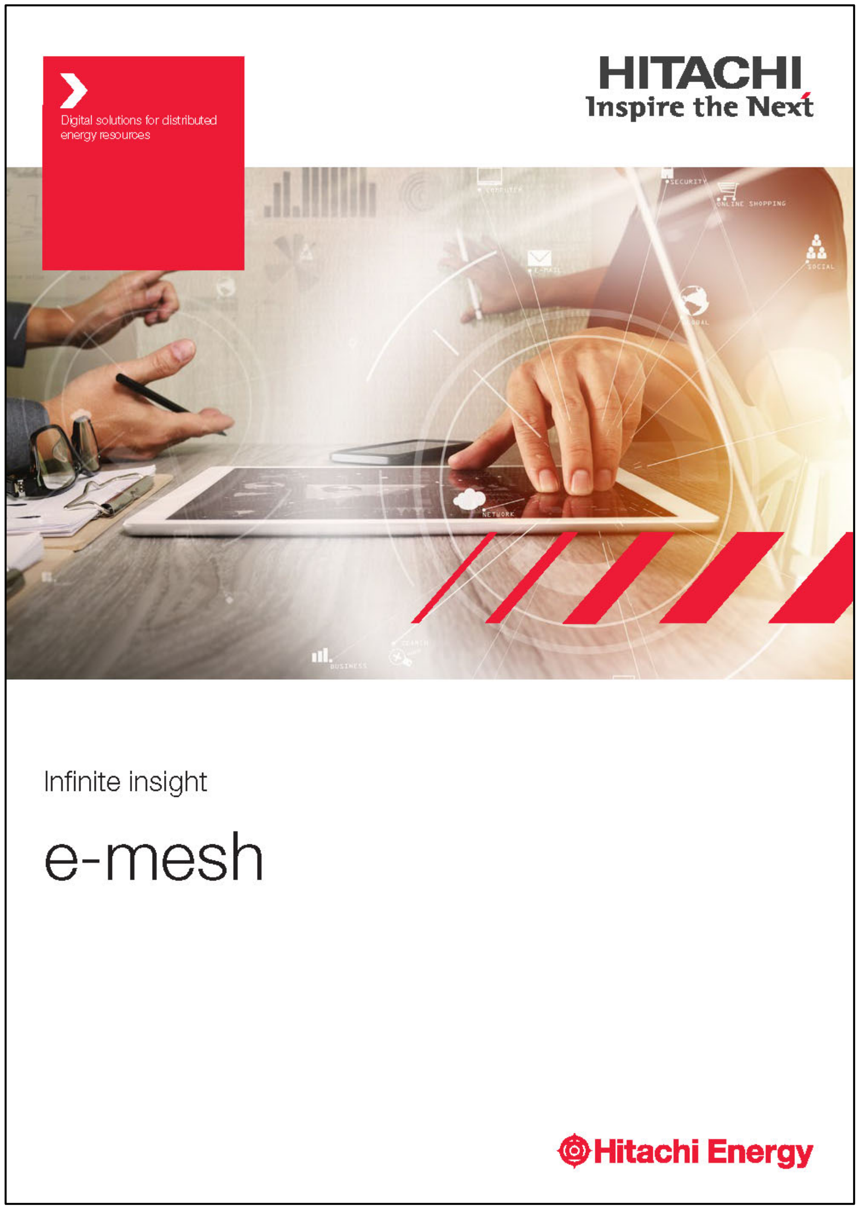
HITACHI
Inspire the Next

Energy Planning & Trading

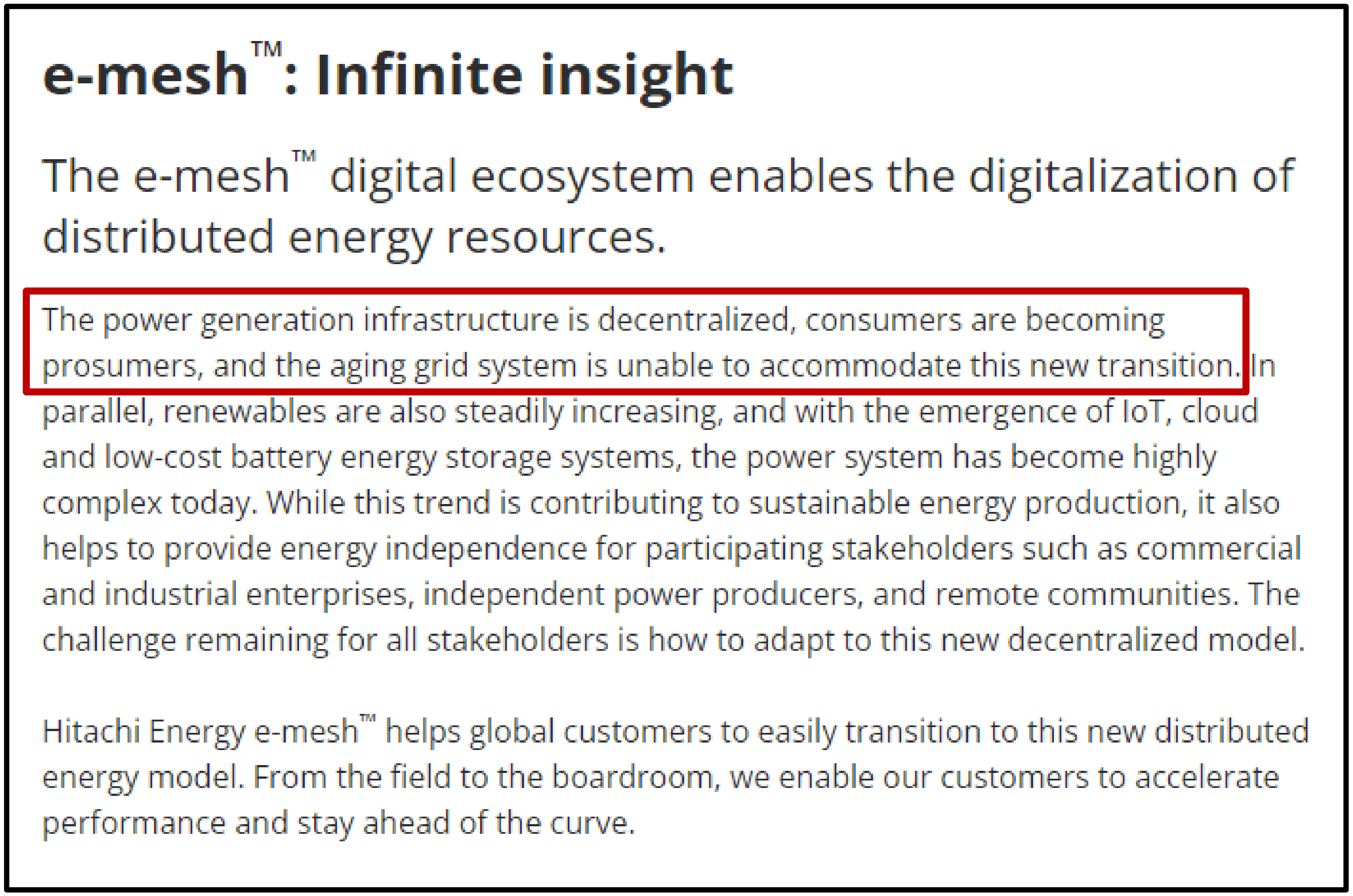
GridView

Hitachi Energy

Case: Hitachi Energy
Managing Distributed Power Grids



Case: Hitachi Energy *Distributed Power Grid Management*



e-mesh™: Infinite insight

The e-mesh™ digital ecosystem enables the digitalization of distributed energy resources.

The power generation infrastructure is decentralized, consumers are becoming prosumers, and the aging grid system is unable to accommodate this new transition. In parallel, renewables are also steadily increasing, and with the emergence of IoT, cloud and low-cost battery energy storage systems, the power system has become highly complex today. While this trend is contributing to sustainable energy production, it also helps to provide energy independence for participating stakeholders such as commercial and industrial enterprises, independent power producers, and remote communities. The challenge remaining for all stakeholders is how to adapt to this new decentralized model.

Hitachi Energy e-mesh™ helps global customers to easily transition to this new distributed energy model. From the field to the boardroom, we enable our customers to accelerate performance and stay ahead of the curve.

Distributed Power

Application

Setting

- ❖ Power grids are changing radically
  - ∗ The infrastructure is decentralizing
  - ∗ Small-scale renewable sources are being added
  - ∗ Low-cost battery storage systems are becoming available
  - ∗ Consumers are becoming power generators, too
- ❖ The current grid system is challenged to adapt

Goal

- ❖ Build a new product for distributed power planning
- ❖ Extend the ABB GridView optimization tools
  to management of renewables and batteries

Evaluation

Only approach considered

- ❖ Extend GridView's AMPL models
  to handle complications posed by new technologies

Choice of AMPL

- ❖ Existing features
  - ∗ Speed, familiarity
  - ∗ Mature APIs
- ❖ Licensing for new deployment formats, such as *containers*

Distributed Power

Sample Formulation

Overview

- Network-constrained unit commitment problem
- Mix of renewable sources & battery storage, within a microgrid

Decision variables

- *Binary:* Whether a plant / battery is on or off at a given time
- *Continuous:* Bi-directional power flows over transmission links

Objective

- Minimize total costs . . .
  - Generation
  - Transmission
  - Supplemental (externally sourced) power
- . . . summed over a given time horizon

Distributed Power

Sample Formulation *(cont'd)*

Objective

- Minimize total costs
  - Generation
  - Transmission
  - Supplemental (externally sourced) power
- summed over a given time horizon

Constraints

- Transmission link capacity
- Battery charge and discharge rates
- Battery capacity
- Demand satisfaction

Distributed Power

Implementation

Project

- Created at ABB Italy (now part of Hitachi Energy)
- 6 project members interacted with AMPL

Tools

- *AMPL Python API*
- VS Code editor for collaboration
- Docker containers, Kubernetes container management
- Amazon Web Services
- D3.js for visualization

Optimization

- Mixed-integer linear solver
- 200K to 45M variables and constraints
- 30 seconds to 20 minutes for each run

Distributed Power

Deployment

Current (2022)

- ❖ Testing with 5-10 commercial clients
- ❖ Expect 30+ adoptions within 2 years

Ongoing benefits

- ❖ Optimization is essential to grid management
- ❖ AMPL enables the core commercial utility of e-mesh

Future possibilities

- ❖ Expand e-mesh's portfolio of features
- ❖ Support vehicle charging and electric fleet optimization

Technology Tutorial:
Optimization in Your Toolchain: How AMPL is Making it Faster and Easier

Tuesday, 9:10-10:00 am

Room Sugarland A

- Expressing constraint logic more directly and understandably
- Exchanging data and results directly and efficiently with spreadsheets and database systems
- Interfacing to business systems through APIs for popular programming languages
- Deploying optimization in cloud environments and containers