

# Developing Optimization Applications Quickly and Effectively with Algebraic Modeling in **AMPL**

*Robert Fourer*

`4er@ampl.com`

AMPL Optimization Inc.

`www.ampl.com` — +1 773-336-AMPL

**INFORMS Annual Meeting**

Houston— 22-25 October 2017

Session MB69, *Technology Tutorials*

# Approaches to Optimization

## *Application-based*

- ❖ Use a software package designed for your problems

## *Method-based*

- ❖ Implement an optimization algorithm for your problems

## ***Model-based***

- ❖ Develop a general description of your problems
- ❖ Send problem instances to an off-the-shelf solver
- ❖ *Compared to application-based:*  
better tailored to your needs
- ❖ *Compared to method-based:*  
much easier to develop and maintain

# The Optimization Modeling Cycle

## *Steps*

- ❖ Communicate with problem owner
- ❖ Build model
- ❖ Prepare data
- ❖ Generate optimization problem
- ❖ Submit problem to solver
- ❖ Report & analyze results
- ❖ ***Repeat!***

## *Goals for optimization modeling software*

- ❖ Do this quickly and reliably
- ❖ Get results before client loses interest
- ❖ ***Deploy for application***

# Optimization Modeling Languages

## *Two forms of an optimization problem*

- ❖ Modeler's form
  - \* Mathematical description, easy for people to work with
- ❖ Algorithm's form
  - \* Explicit data structure, easy for solvers to compute with

## *Idea of a modeling language*

- ❖ ***A computer-readable modeler's form***
  - \* You write optimization problems in a modeling language
  - \* Computers translate to algorithm's form for solution

## *Advantages of a modeling language*

- ❖ Faster modeling cycles
- ❖ More reliable modeling
- ❖ More maintainable applications

# Algebraic Modeling Languages

## *Formulation concept*

- ❖ Define data in terms of sets & parameters
  - \* Analogous to database keys & records
- ❖ Define decision variables
- ❖ Minimize or maximize a function of decision variables
- ❖ Subject to equations or inequalities that constrain the values of the variables

## *Advantages*

- ❖ Familiar
- ❖ Powerful
- ❖ Proven

# Categorizations of Algebraic Modeling Languages

## *By language design*

- ❖ Extended from a general programming language
- ❖ Built specially for optimization

## *By solver support*

- ❖ Specialized for one particular solver
- ❖ Designed to support many solvers



## *Features*

- ❖ Algebraic modeling language
- ❖ Built specially for optimization
- ❖ Designed to support many solvers

## *Design goals*

- ❖ Powerful, general expressions
- ❖ Natural, easy-to-learn modeling principles
- ❖ Efficient processing that scales well with problem size

***4 ways to use . . .***

# 4 Ways to Use AMPL

## *Command language*

- ❖ Browse results & debug model interactively
- ❖ Make changes and re-run

## *Scripting language*

- ❖ Bring the programmer to the modeling language

## *Programming interface (API)*

- ❖ Bring the modeling language to the programmer

## *Deployment tool (QuanDec)*

- ❖ Embed models into an interactive decision-making tool



# Example

## *Roll cutting model*

- ❖ Solution via command language
- ❖ Tradeoff analysis via scripting

## *Roll cutting by pattern enumeration*

- ❖ via scripting
- ❖ via API

## *Roll cutting by pattern generation*

- ❖ via scripting
- ❖ via API

*. . . featuring new AMPL API for Python*

## *In practice . . .*

### *A general tool for applying optimization*

- ❖ Based on a broadly applicable paradigm
- ❖ Readily accommodates unanticipated requirements

### *Ideally positioned for new projects*

- ❖ More control
  - \* compared to application-specific software
- ❖ Faster, more flexible prototyping
  - \* compared to development in a programming language

### *Scalable for integration and deployment*

# Roll Cutting Problem

## *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  $j = 1, \dots, n$

$b_i$  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  $j = 1, \dots, n$

*to minimize*

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

total number of rolls cut

*subject to*

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

number of rolls of width  $i$  cut  
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];
```

$$\sum_{j=1}^n a_{ij} X_j \geq b_i$$

*Roll Cutting*

# 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 ;
```

## *In practice . . .*

### *Model: decision variables, objective, constraints*

- ❖ Applicable for many problem types
  - \* Planning, scheduling, routing, packing, assignment
  - \* Network flow, portfolio selection, feedstock blending
- ❖ Successful in many business areas
  - \* Production, logistics, sequencing, assignment, design
  - \* Energy, manufacture, process, finance, commerce

### *Model + data = Optimization problem for solver*

- ❖ Model defined & documented independently of data
- ❖ Varied data sources supported
  - \* Text files, spreadsheets, databases, API calls



# Command Language

*Model + data = problem instance to be solved*

```
ampl: model cut.mod;
ampl: data cut.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 12.7.1.0: 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 7.5.0: 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, 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 {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      # pattern
2 61.75   4 61.24   6 62.54   8 62.0          # total widths

ampl: display Fulfill.slack;
6.77  2                                # overruns
7.56  3
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];
```

*Roll Cutting*

# 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 7.5.0: optimal solution; objective 20
3 simplex iterations
ampl: display Number, Waste;
Number = 20
Waste = 63.62

ampl: objective Waste; solve;
Gurobi 7.5.0: optimal solution; objective 15.62
2 simplex iterations
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 := 6 ;

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 7.5.0: optimal solution; objective 20
8 simplex iterations
1 branch-and-cut nodes

ampl: display Number, Waste;
Number = 20
Waste = 54.76

ampl: objective Waste; solve;
Gurobi 7.5.0: optimal solution; objective 49.16
4 simplex iterations

ampl: display Number, Waste;
Number = 21
Waste = 49.16
```

# Further revisions

## *Overruns*

- ❖ Limit to percentage of amount ordered
- ❖ Limit total extra rolls

## *Pattern restrictions*

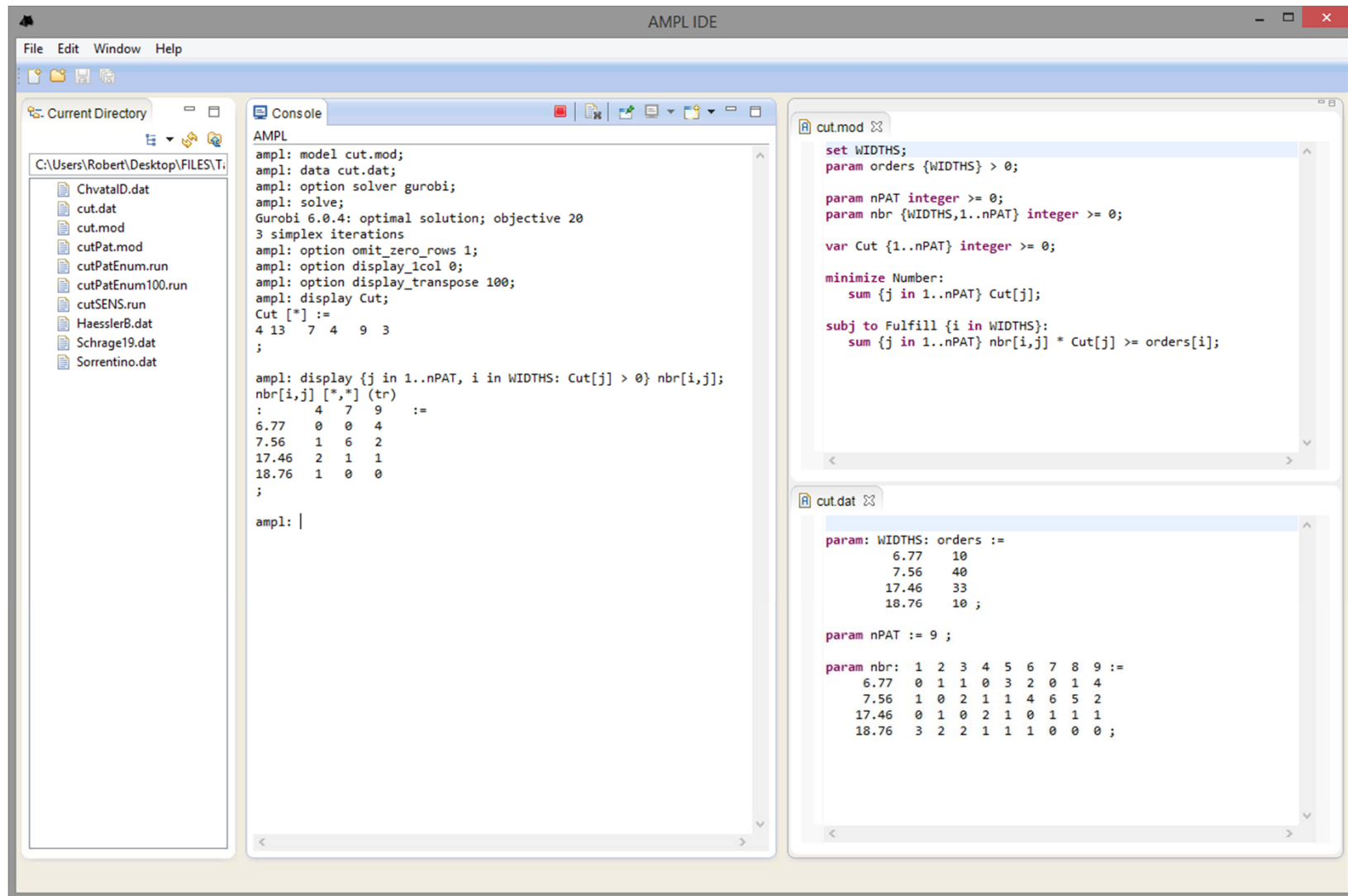
- ❖ Cut at least a specified number of each pattern used
- ❖ Limit the number of patterns used

## *Costs*

- ❖ Account for setups
- ❖ Account for complications of cutting

*Anything else you can imagine . . .*

# IDE for Command Language



The screenshot displays the AMPL IDE interface. On the left is a file explorer showing the current directory: C:\Users\Robert\Desktop\FILES\T. The central console window shows the execution of an AMPL model, including the command 'ampl: solve;' and the output 'Gurobi 6.0.4: optimal solution; objective 20'. The console also displays the results of 'ampl: display Cut;' and 'ampl: display {j in 1..nPAT, i in WIDTHS: Cut[j] > 0} nbr[i,j];'. On the right, two code editors are visible. The top editor, 'cut.mod', contains the AMPL model code, including parameter declarations for WIDTHS, nPAT, and nbr, and a minimize objective function. The bottom editor, 'cut.dat', contains the data for the model, including the WIDTHS parameter values and the nbr matrix.

```
File Edit Window Help
Current Directory
C:\Users\Robert\Desktop\FILES\T
ChvatalD.dat
cut.dat
cut.mod
cutPat.mod
cutPatEnum.run
cutPatEnum100.run
cutSENS.run
HaesslerB.dat
Schrage19.dat
Sorrentino.dat

Console
AMPL
ampl: model cut.mod;
ampl: data cut.dat;
ampl: option solver gurobi;
ampl: solve;
Gurobi 6.0.4: optimal solution; objective 20
3 simplex iterations
ampl: option omit_zero_rows 1;
ampl: option display_1col 0;
ampl: option display_transpose 100;
ampl: display Cut;
Cut [*] :=
4 13 7 4 9 3
;

ampl: display {j in 1..nPAT, i in WIDTHS: Cut[j] > 0} nbr[i,j];
nbr[i,j] [*,*] (tr)
:=
6.77 0 0 4
7.56 1 6 2
17.46 2 1 1
18.76 1 0 0
;

ampl: |

cut.mod
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];

cut.dat
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 ;
```

## *In practice . . .*

### *Work interactively*

- ❖ Make changes
- ❖ Solve
- ❖ Browse results
- ❖ *Review and repeat*

### *Choose the best solver for your problem*

- ❖ Linear/quadratic mixed-integer
  - \* CPLEX, Gurobi, Xpress
- ❖ Nonlinear continuous
  - \* CONOPT, Ipopt, LGO, LOQO, MINOS, SNOPT
- ❖ Nonlinear mixed-integer
  - \* BARON, Bonmin, Couenne, Knitro

# Scripting

*Bring the programmer to the modeling language*

*Extend 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 objectives
- ❖ Cutting *via* pattern enumeration
- ❖ Cutting *via* pattern generation

*Scripting*

## **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];
```



*Scripting*

# Parametric Analysis (*cont'd*)

*Script run*

```
ampl: include cutWASTE.run
```

```
Min 20 rolls with waste 63.62
```

Over	Waste	Number
10	46.72	22
7	47.89	21
5	54.76	20

```
ampl:
```

*Scripting*

# 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];
```

*Scripting*

# 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;
data Sorrentino.dat;

param curr_sum >= 0;
param curr_width > 0;
param pattern {WIDTHS} integer >= 0;

let maxPAT := 1000000;

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

*Scripting*

# 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)*

```
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";
}
printf "\nWASTE = %5.2f%\n\n",
    100 * (1 - (sum {i in WIDTHS} i * orders[i]) / (roll_width * Number));
```

*Scripting*

# Pattern Enumeration

## *Results*

```
ampl: include cutPatEnum.run
```

```
Gurobi 7.5.0: optimal solution; objective 18
```

```
9 simplex iterations
```

```
1 branch-and-cut node
```

```
43 patterns, 18 rolls
```

Cut	3	1	3	11
18.76	3	1	0	0
17.46	0	2	3	2
7.56	1	1	1	3
6.77	0	0	0	1

```
WASTE = 2.34%
```



*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
```

```
Gurobi 7.5.0: optimal solution; objective 34  
130 simplex iterations
```

```
54508 patterns, 34 rolls
```

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

```
WASTE = 0.69%
```

*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 a subset of patterns)*

```
ampl: include cutPatEnum.run
```

```
Gurobi 7.5.0: optimal solution; objective 33
```

```
362 simplex iterations
```

```
1 branch-and-cut nodes
```

```
273380 patterns, 33 rolls
```

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

```
WASTE = 0.62%
```

*Scripting*

## **Cutting *via* Pattern Generation**

*Generate the pattern list by a series of solves*

- ❖ Solve LP relaxation using subset of patterns
- ❖ Add “most promising” pattern to the subset
  - \* Minimize reduced cost given dual values
  - \* Equivalent to a knapsack problem
- ❖ Iterate as long as there are promising patterns
  - \* Stop when minimum reduced cost is zero
- ❖ Solve IP using all patterns found

*Scripting*

# Pattern Generation

## *Cutting model*

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

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

*Scripting*

# Pattern Generation

## *Knapsack model*

```
param roll_width > 0;
param price {WIDTHS} default 0.0;

var Use {WIDTHS} integer >= 0;

minimize Reduced_Cost:
    1 - sum {i in WIDTHS} price[i] * Use[i];

subj to Width_Limit:
    sum {i in WIDTHS} i * Use[i] <= roll_width;
```

*Scripting*

# Pattern Generation

*Script (problems, initial patterns)*

```
model cutPatGen.mod;
data Sorrentino.dat;

problem Cutting_Opt: Cut, Number, Fill;
    option relax_integrality 1;
    option presolve 0;

problem Pattern_Gen: Use, Reduced_Cost, Width_Limit;
    option relax_integrality 0;
    option presolve 1;

let nPAT := 0;
for {i in WIDTHS} {
    let nPAT := nPAT + 1;
    let nbr[i,nPAT] := floor (roll_width/i);
    let {i2 in WIDTHS: i2 <> i} nbr[i2,nPAT] := 0;
};
```



*Scripting*

# Pattern Generation

*Script (generation loop)*

```
repeat {
  solve Cutting_Opt;
  let {i in WIDTHS} price[i] := Fill[i].dual;
  solve Pattern_Gen;
  printf "\n%7.2f%11.2e  ", Number, Reduced_Cost;
  if Reduced_Cost < -0.00001 then {
    let nPAT := nPAT + 1;
    let {i in WIDTHS} nbr[i,nPAT] := Use[i];
  }
  else break;
  for {i in WIDTHS} printf "%3i", Use[i];
};
```

*Scripting*

# Pattern Generation

*Script (final integer solution)*

```
option Cutting_Opt.relax_integrality 0;
option Cutting_Opt.presolve 10;
solve Cutting_Opt;

if Cutting_Opt.result = "infeasible" then
    printf "\n*** No feasible integer solution ***\n\n";
else {
    printf "Best integer: %3i rolls\n\n", sum {j in 1..nPAT} Cut[j];
    for {j in 1..nPAT: Cut[j] > 0} {
        printf "%3i of:", Cut[j];
        printf {i in WIDTHS: nbr[i,j] > 0}: "%3i x %6.3f", nbr[i,j], i;
        printf "\n";
    }

    printf "\nWASTE = %5.2f%%\n\n",
        100 * (1 - (sum {i in WIDTHS} i * orders[i]) / (roll_width * Number));
}
```

*Scripting*

# Pattern Generation

*Results (relaxation)*

```
ampl: include cutpatgen.run
```

```
20.44 -1.53e-01 1 3 2 0
18.78 -1.11e-01 0 1 3 0
18.37 -1.25e-01 0 1 0 3
17.96 -4.17e-02 0 6 0 1
17.94 -1.00e-06
```

Optimal relaxation: **17.9412 rolls**

```
10.0000 of: 1 x 6.770 3 x 7.560 2 x 17.460
4.3333 of: 1 x 7.560 3 x 17.460
3.1961 of: 1 x 7.560 3 x 18.760
0.4118 of: 6 x 7.560 1 x 18.760
```

WASTE = 2.02%

*Scripting*

# Pattern Generation

*Results (integer)*

Rounded up to integer: **20 rolls**

Cut	10	5	4	1
6.77	1	0	0	0
7.56	3	1	1	6
17.46	2	3	0	0
18.76	0	0	3	1

WASTE = 12.10%

Best integer: **19 rolls**

Cut	10	5	3	1
6.77	1	0	0	0
7.56	3	1	1	6
17.46	2	3	0	0
18.76	0	0	3	1

WASTE = 7.48%

## *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*

- ❖ Close AMPL environment before returning to system

*So . . .*

# 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

*Add-ons to all AMPL distributions*

- ❖ Java, MATLAB, C++, C#
  - \* Download from <http://ampl.com/products/api/>
- ❖ **Python** 2.7, 3.3, 3.4, 3.5, 3.6
  - \* `pip install amplpy`

# Cutting Revisited

## *Hybrid approach*

- ❖ Control & pattern creation from a programming language
  - \* Pattern enumeration: finding all patterns
  - \* Pattern generation: solving knapsack problems
- ❖ Model & modeling commands in AMPL

## *Key to Python program examples*

- ❖ AMPL entities
- ❖ AMPL API Python objects
- ❖ AMPL API Python methods
- ❖ Python functions etc.



# AMPL Model File

## *Basic pattern-cutting model*

```
param nPatterns integer > 0;

set PATTERNS = 1..nPatterns; # patterns
set WIDTHS; # finished widths

param order {WIDTHS} >= 0; # rolls of width j ordered
param overrun; # permitted overrun on any width

param rawWidth; # width of raw rolls to be cut
param rolls {WIDTHS,PATTERNS} >= 0, default 0; # rolls of width i in pattern j

var Cut {PATTERNS} integer >= 0; # raw rolls to cut in each pattern

minimize TotalRawRolls: sum {p in PATTERNS} Cut[p];

subject to FinishedRollLimits {w in WIDTHS}:
    order[w] <= sum {p in PATTERNS} rolls[w,p] * Cut[p] <= order[w] + overrun;
```

*AMPL API*

## Some Python Data

*A float, an integer, and a dictionary*

```
roll_width = 64.5
overrun = 6
orders = {
    6.77: 10,
    7.56: 40,
    17.46: 33,
    18.76: 10
}
```

*... can also work with  
lists and Pandas dataframes*

# 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))
}
```

# 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
```

# Pattern Enumeration in Python

## *Plotting routine*

```
def cuttingPlot(roll_width, widths, summary, 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']
```



# 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"
        )

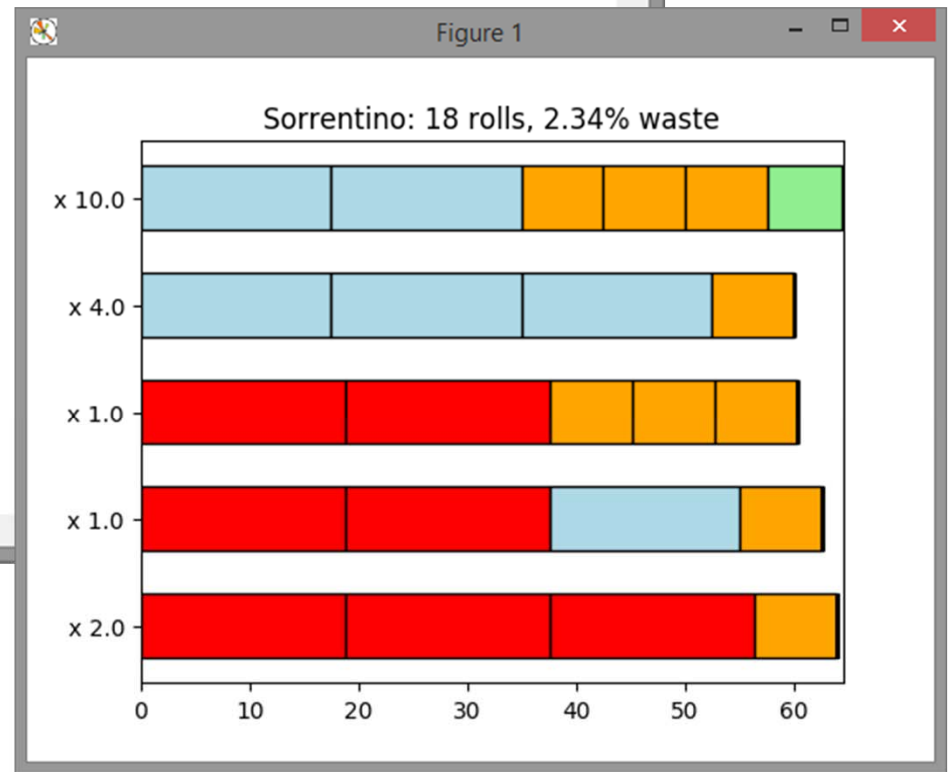
plt.xlim(0, roll_width)
plt.xticks(np.arange(0, roll_width, 10))
plt.yticks(ind, tuple("x {}".format(rep) for patt, rep in solution))

plt.show()
```

# Pattern Enumeration in Python

```
2017: running ipython
File Edit Help
sw: ipython
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct 6 2014, 22:16:31) [MSC v.1600 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 6.1.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from pattern_enumeration import *
In [2]: cuttingEnum('Sorrentino')
43 patterns generated
Gurobi 7.5.0: optimal solution; objective 18
9 simplex iterations
1 branch-and-cut nodes
```



## *In practice . . .*

*Integrate within a larger scheme*

*Retain benefits of algebraic modeling*

- ❖ work with natural representation of optimization models
- ❖ efficient prototyping, reliable maintenance

*Use the best tools for each part of the project*

- ❖ program data manipulation in your choice of language
- ❖ work with optimization models in AMPL

# Pattern Generation in Python

*Get data, set up master problem*

```
function cuttingGen(dataset)
  from amplpy import AMPL

  # Read orders, roll_width, overrun; extract widths
  exec(open(dataset+'.py').read(), globals())
  widths = list(sorted(orders.keys(), reverse=True))

  # Set up cutting (master problem) model
  Master = AMPL()
  Master.option['ampl_include'] = 'models'
  Master.read('cut.mod')

  # Define a param for sending new patterns
  Master.eval('param newPat {WIDTHS} integer >= 0;')

  # Set solve options
  Master.option['solver'] = 'gurobi'
  Master.option['relax_integrality'] = 1
```

# Pattern Generation in Python

## *Send data to master problem*

```
# Send scalar values
Master.param['nPatterns'] = len(widths)
Master.param['overrun'] = overrun
Master.param['rawWidth'] = roll_width

# Send order vector
Master.set['WIDTHS'] = widths
Master.param['order'] = orders

# Generate and send initial pattern matrix
Master.param['rolls'] = {
    (widths[i], 1+i): int(floor(roll_width/widths[i]))
    for i in range(len(widths))
}
```

# Pattern Generation in Python

## *Set up subproblem*

```
# Define knapsack subproblem
Sub = AMPL()
Sub.option['solver'] = 'gurobi'
Sub.eval('''
    set SIZES;
    param cap >= 0;
    param val {SIZES};
    var Qty {SIZES} integer >= 0;
    maximize TotVal: sum {s in SIZES} val[s] * Qty[s];
    subject to Cap: sum {s in SIZES} s * Qty[s] <= cap;
''')

# Send subproblem data
Sub.set['SIZES'] = widths
Sub.param['cap'] = roll_width
```

# Pattern Generation in Python

*Generate patterns and re-solve cutting problems*

```
# Alternate between master and sub solves
while True:
    Master.solve()

    Sub.param['val'].setValues(Master.con['OrderLimits'].getValues())
    Sub.solve()
    if Sub.obj['TotVal'].value() <= 1.00001:
        break

    Master.param['newPat'].setValues(Sub.var['Qty'].getValues())
    Master.eval('let nPatterns := nPatterns + 1;')
    Master.eval('let {w in WIDTHS} rolls[w, nPatterns] := newPat[w];')

# Compute integer solution
Master.option['relax_integrality'] = 0
Master.solve()
```

# Pattern Generation in Python

## *Display solution*

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

# Retrieve patterns and solution
npatterns = int(Master.param['nPatterns'].value())
rolls = Master.param['rolls'].getValues().toDict()
cutvec = Master.var['Cut'].getValues().toDict()
```



# Pattern Generation in Python

## *Display solution*

```
# Prepare solution data
solution = [
    ([int(rolls[widths[i], p+1][0])
     for i in range(len(widths))], int(cutvec[p+1][0]))
    for p in range(npatterns)
    if cutvec[p+1][0] > 0
]

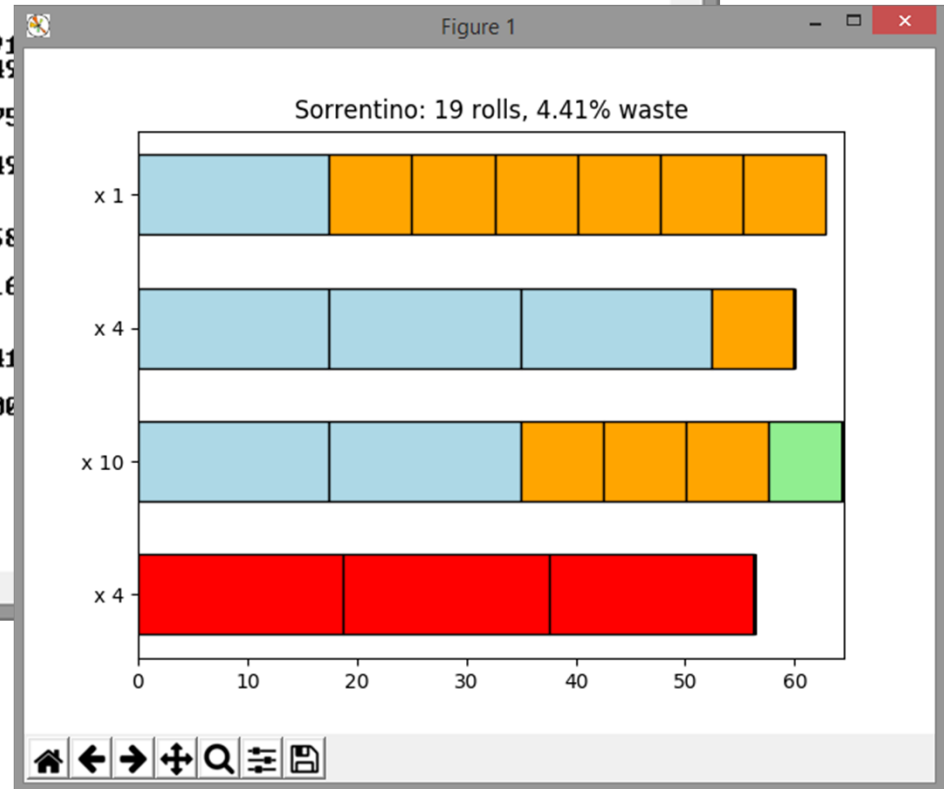
# Create plot of solution
cuttingPlot(roll_width, widths, summary, solution)
```

# Pattern Generation in Python

```
sw: running ipython
File Edit Help
sw: ipython --no-banner

In [1]: from pattern_generation import *

In [2]: cuttingGen('Sorrentino')
Gurobi 7.5.0: optimal solution; objective 20.44444444
Gurobi 7.5.0: optimal solution; objective 1.152777
2 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 18.791
Gurobi 7.5.0: optimal solution; objective 1.1249
1 simplex iterations
Gurobi 7.5.0: optimal solution; objective 18.375
3 simplex iterations
Gurobi 7.5.0: optimal solution; objective 1.1249
1 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 17.958
5 simplex iterations
Gurobi 7.5.0: optimal solution; objective 1.0416
5 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 17.941
5 simplex iterations
Gurobi 7.5.0: optimal solution; objective 1.0000
1 simplex iterations
1 branch-and-cut nodes
Gurobi 7.5.0: optimal solution; objective 19
3 simplex iterations
1 branch-and-cut nodes
```



## *In practice . . .*

### *Implement hybrid iterative schemes*

- ❖ build powerful software for hard problems

### *Alternate between optimization & other analytics*

- ❖ invoke specialized optimizers for subproblems

# QuanDec

## *Server side*

- AMPL model and data
- Standard AMPL-solver installations

## *Client side*

- Interactive tool for collaboration & decision-making
- Runs on any recent web browser
- Java-based implementation
  - \* AMPL API for Java
  - \* Eclipse Remote Application Platform

*. . . developed / supported by Cassotis Consulting*

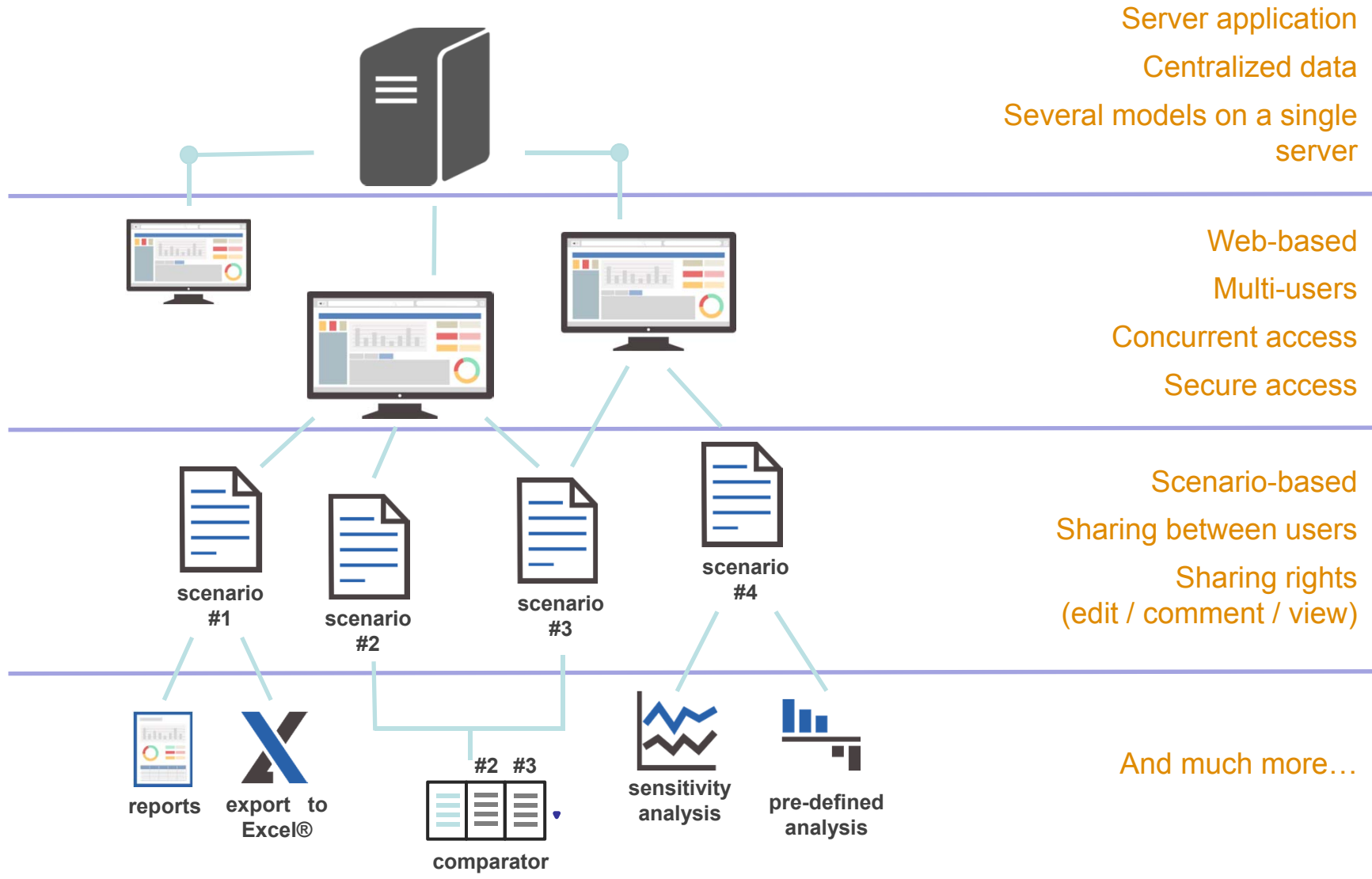
# Quan $\Delta$ ec



The web-based graphical interface that turns optimization models written in AMPL into decision-making tools.



# Features

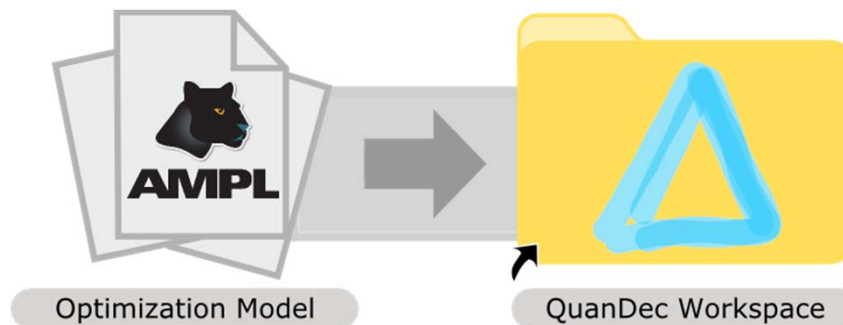


# Getting started

**step 1:** install QuanDec on a server

**step 2:** copy & paste your model files (.mod and .dat) into  
QuanDec's workspace

**step 3:** create AMPL tables and link them to QuanDec explorer





E-mail :

Password :

[Forgot?](#)

Enter your email to login

Version 2.3.1

**CASSOTIS** consulting

Login

**Web-application**

**Multi-user**


**Secure access**

**Concurrent access**





Workspace Admin





Switch workspace New Master Import Master Compare

Quan  ec

This week

Name	Owner	Last change
 BUDGET 2016	Mary Torres	September 9, 2016 4:59 PM
 My Scenario	Me	Today 10:54 AM



All

Name	Owner
 BUDGET 2015	Mary Torres
 BUDGET 2016	Mary Torres
 My Scenario	Me
 FORECAST 2017	Mary Torres

Share with others

Anyone can comment

People or groups

 Robert Finn can edit 

OK

**Scenario-based environment**

**Sharing system**

**Permission:  
Edit – Comment - View**

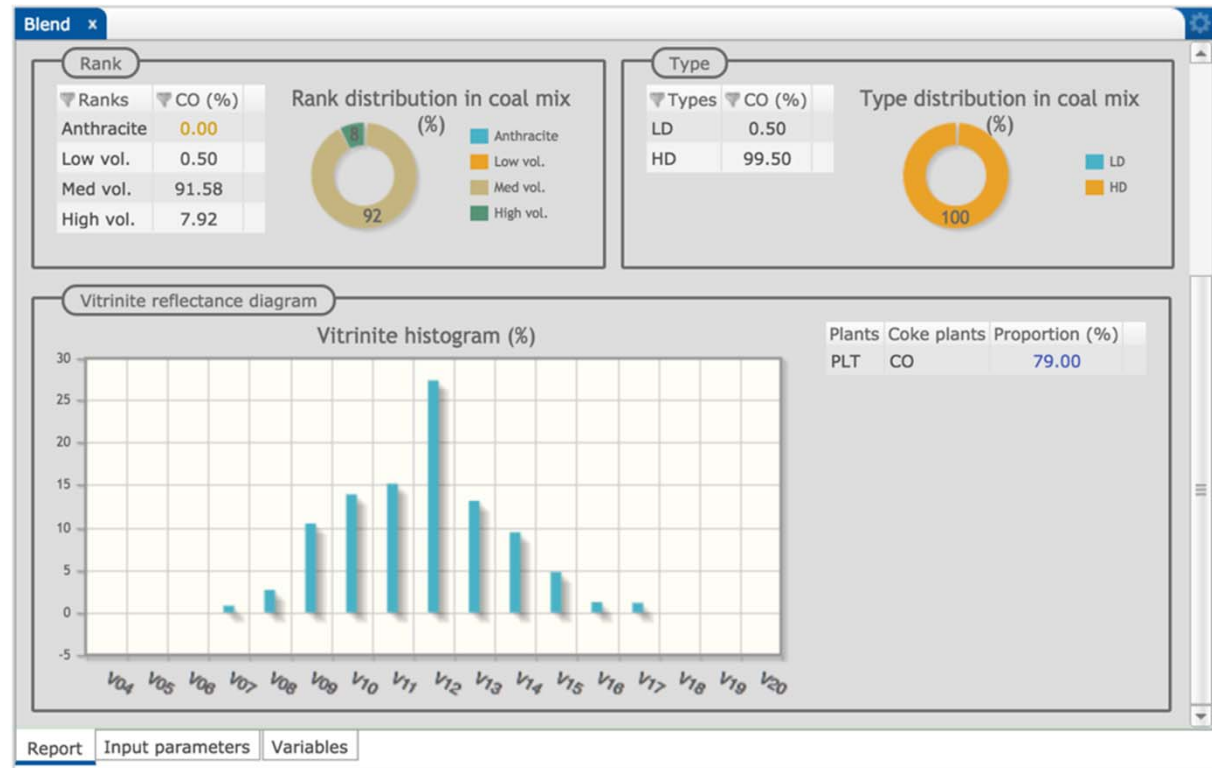
### 3 levels:

- Report
- Input parameters
- Variables

Chart and tables

Colored values  
for easier analysis

Constraint (min/max)  
on any variable



Collaborative work

Notification system

Comments between users

The screenshot displays a software interface with several components:

- Table:** A table with columns for 'blend', 'SM1 (kg/t)', and 'WAVG (kg/t)'.

blend	SM1 (kg/t)	WAVG (kg/t)
Hot metal	889.89	889.89
Lump ores	0.00	0.00
Pellets	0.00	0.00
- Chart:** A horizontal bar chart titled 'Blend at conver' showing the composition of materials. The x-axis ranges from 0 to 1200. The legend includes Hot metal (blue), Lump ores (orange), Pellets (yellow), Recyclings (green), Fluxes (dark green), and Ferroalloys (brown).
- Modal Windows:**
  - Comment this value:** A dialog box with a dropdown menu set to 'QUESTION'. It contains two comments: one from Mary Torres asking 'Do we use pellets at the converter?' and one from Benjamin Steward replying 'No, we exclusively use lump ores.' There is an 'Enter a message:' field and 'Cancel'/'OK' buttons.
  - Share with others:** A dialog box with a dropdown menu set to 'Anyone can comment'. It lists 'Robert Finn' as a user who 'can edit' with a red 'X' icon. There is an 'OK' button.
  - New comment:** A notification bubble in the top right corner stating 'Robert Finn has commented this dataset.'

Coke plants x

Operating costs

Plants	Coke plants	Costs	Fixed (MUS\$/year)	Variable (US\$/t)
PLT	CO	Maintenance	7.75	0.90
PLT	CO	Labour costs	3.95	0.00
PLT	CO	Utilities	0.05	0.11
PLT	CO	Water treatment	7.78	0.00
PLT	CO	Court yard	5.36	0.00
PLT	CO	Services	0.02	0.94
PLT	CO	Indirect costs	2.57	0.00
PLT	CO	Depreciation	4.92	0.00
PLT	CO	Electricity	0.00	0.03

Report Input parameters Variables

Journal Bounds Regressions Comments Error Log

Journal	Bounds	Regressions	Comments	Error Log
Operating cost at coke plant	PLT, CO1, co_elec, Variable	0.03	Today 11:26 AM	by Arthur Turner
CO operational costs	PLT, co_elec	Electricity	Today 11:26 AM	by Arthur Turner
CO operational costs	PLT	co_elec	Today 11:26 AM	by Arthur Turner
Vitrinite reflectance inside of range at coke plant	PLT, CO1	MAX 79.00	Today 10:49 AM	by Arthur Turner

Arthur Turner QuanDec STEEL BUDGET 2016 My Scenario

Scenarios with changes history

Traceability and undo system

Workspace Admin

New Report Show/Hide differences Export to Excel

Quan<sup>Δ</sup>ec

Comparator

Variable	Unit	BUDGET 2016	My Scenario	Diff
Executive summaries				
Costs and Revenues				
Profit and Sales				
Production costs				
Absolute costs	MUS\$			
Detailed costs	US\$/t			
Internal price of intermedi	US\$/t			
Net production level	kt			
'PLT' 'CO'	kt	1763.98	1764.25	0.02%
'PLT' 'SI'	kt	4085.77	4084.46	-0.03%
'PLT' 'BF'	kt	5062.62	5060.91	-0.03%
'PLT' 'ST'	kt	5258.29	5256.75	-0.03%
'PLT' 'PO'				
Production cost of prod				
Production level				
Material blends				
Coke plants				
Sinter plants				
Blast furnaces				
Steel shops				
Power plant				
Raw materials				

Select the scenarios to compare:

- BUDGET 2015
- BUDGET 2016
- My Scenario
- FORECAST 2017

Cancel OK

Economics and Production

Variable	Index	Unit	BUDGET 2016	My Scenario	Diff
Economics per int. plant	'PLT' 'costs'	MUS\$	1515.59	1515.20	-0.03%
Economics per int. plant	'PLT' 'revenues'	MUS\$	1762.23	1761.77	-0.03%
Economics per int. plant	'PLT' 'profit'	MUS\$	246.64	246.56	-0.03%
Economics per int. plant	'PLT' 'margin'	%	14.00	14.00	-0.00%
Production cost of product	'PLT' 'coke'	US\$/t	164.48	164.54	0.04%
Production cost of product	'PLT' 'sinter'	US\$/t	77.55	77.50	-0.06%
Production cost of product	'PLT' 'hotmetal'	US\$/t	193.95	193.99	0.02%
Production cost of product	'PLT' 'slab'	US\$/t	286.27	286.28	0.00%
Production cost of product	'PLT' 'electricity'	US\$/MWh	125.75	125.75	0.00%
Production level of product	'PLT' 'coke'	kt	1818.54	1818.81	0.02%
Production level of product	'PLT' 'sinter'	kt	4085.77	4084.46	-0.03%

Report Structure

Reports

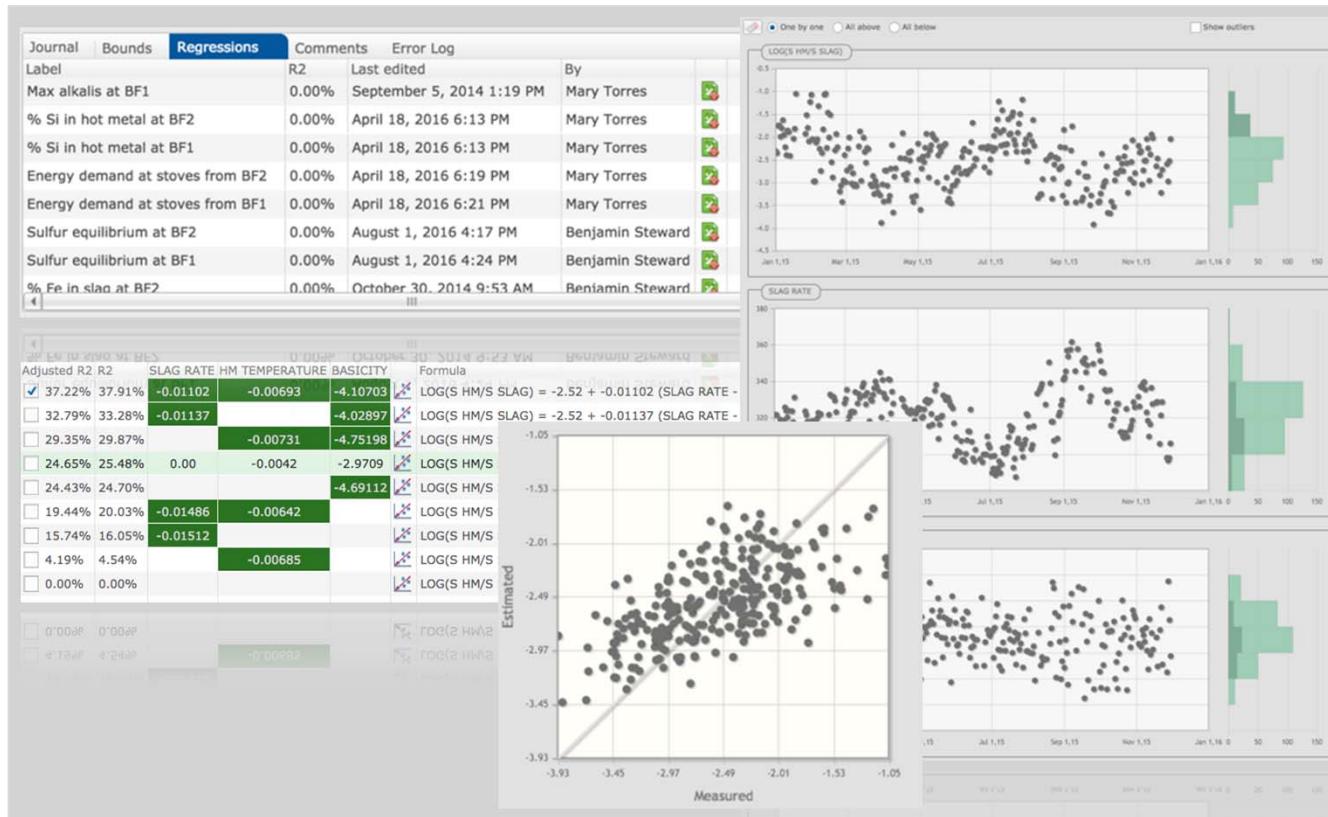
Name	User	Date	Action
Sulfur cycle	Benjamin Steward	March 18, 2016 3:45 PM	✖
Metallic blend at CV	Me	February 21, 2016 4:51 PM	✖
Raw material use at Reduction	Me	January 15, 2016 4:36 PM	✖
Economics and Production	Mary Torres	September 13, 2016 4:53 PM	✖
Flux consumption at Torpedo	Mary Torres	April 3, 2016 4:44 PM	✖
Slab sales	Robert Finn	January 30, 2016 5:30 PM	✖
Silicon cycle	Benjamin Steward	July 5, 2016 4:17 PM	✖

Scenario comparison

All variables can be compared

Display of relative difference

Custom reports



Regression tool

Data cleaning

Any variable can be added to a regression

Manual coefficients if no data available

Profit and Sales

Exchange rates

Currencies	Exchange rates (US\$)
eur	1.14
usd	1.00
brl	0.29

Horizon

Days (d) 365.00

- Add a comment
- Add a constraint
- Add to a regression
- Analyse sensitivity
- Export the table
- Download the template

Sensitivity analysis

Parameter : Exchange rates

Index : 'brl'

From : 0.3

To : 1

# Pts : 3

Cancel OK

Sensitivity analysis

For both parameters AND variables

All variables can be compared

Display of relative difference

Workspace Admin

Back to edition New Report Show/Hide differences Export to Excel

QuanDec

Comparator

Variable	Unit	0.30	0.65	Diff	1.00	Diff
Executive summaries						
Costs and Revenues						
Profit and Sales						
Economics per int. plant	MUS\$					
'PLT' 'costs'	MUS\$	1515.39	1544.99	1.95%	1633.34	7.78%
'PLT' 'revenues'	MUS\$	1754.70	1679.96	-4.26%	1670.71	-4.79%
'PLT' 'profit'	MUS\$	239.31	134.97	-43.60%	37.37	-84.38%
'PLT' 'margin'	%	13.64	8.03	-41.09%	2.24	-83.60%
Global economics	MUS\$					
External costs per process	MUS\$					
External costs per type	MUS\$					
Detailed external costs	MUS\$					
External revenues per process	MUS\$					
External revenues per type	MUS\$					
Detailed external revenues	MUS\$					
Detailed revenues	MUS\$/t					
Production costs						
Material blends						
Coke plants						
Sinter plants						
Blast furnaces						
Steel shops						
Power plant						
Raw materials						
Gases						

Economics and Production

Variable	Index	Unit	0.30	0.65	Diff	1.00	Diff
Economics per int. plant	'PLT' 'costs'	MUS\$	1515.39	1544.99	1.95%	1633.34	7.78%
Economics per int. plant	'PLT' 'revenues'	MUS\$	1754.70	1679.96	-4.26%	1670.71	-4.79%
Economics per int. plant	'PLT' 'profit'	MUS\$	239.31	134.97	-43.60%	37.37	-84.38%
Economics per int. plant	'PLT' 'margin'	%	13.64	8.03	-41.09%	2.24	-83.60%
Production cost of product	'PLT' 'coke'	US\$/t	164.51	161.52	-1.82%	162.71	-1.10%
Production cost of product	'PLT' 'sinter'	US\$/t	77.68	83.23	7.15%	88.16	13.50%
Production cost of product	'PLT' 'hotmetal'	US\$/t	194.23	198.43	2.16%	202.93	4.48%
Production cost of product	'PLT' 'slab'	US\$/t	287.62	307.33	6.85%	326.85	13.64%
Production cost of product	'PLT' 'electricity'	US\$/MWh	125.62	125.73	0.08%	125.74	0.09%
Production level of product	'PLT' 'coke'	kt	1818.81	1815.95	-0.16%	1815.95	-0.16%
Production level of product	'PLT' 'sinter'	kt	4115.36	4007.25	-2.63%	4006.24	-2.65%
Production level of product	'PLT' 'hotmetal'	kt	5105.94	5051.71	-1.06%	5052.00	-1.06%
Production level of product	'PLT' 'trhotmetal'	kt	5025.36	4972.09	-1.06%	4972.37	-1.05%
Production level of product	'PLT' 'crudsteel'	kt	5657.39	5402.17	-4.51%	5372.49	-5.04%

Report Structure

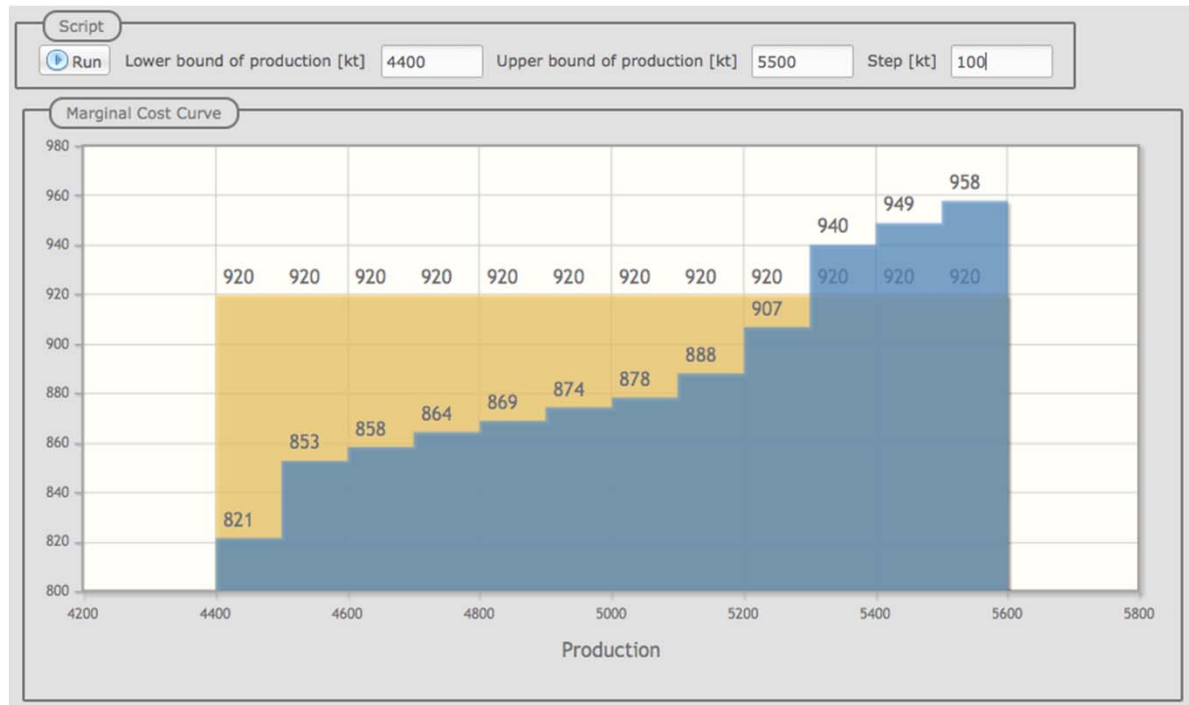
Reports

Name	User	Date	Action
Sulfur cycle	Benjamin Steward	March 18, 2016 3:45 PM	✖
Metallic blend at CV	Me	February 21, 2016 4:51 PM	✖
Raw material use at Reduction	Me	January 15, 2016 4:36 PM	✖
Economics and Production	Mary Torres	September 13, 2016 4:53 PM	✖
Flux consumption at Torpedo	Mary Torres	April 3, 2016 4:44 PM	✖
Slab sales	Robert Finn	January 30, 2016 5:30 PM	✖
Silicon cycle	Benjamin Steward	July 5, 2016 4:17 PM	✖

Arthur Turner QuanDec STEEL BUDGET 2016 My Scenario About QuanDec...

Predefined analyses

Script parameters





# QuanDec Availability

*Ready now for commercial applications*

- Free trials available
- Pricing keyed to number of models & users

*First year's support included*

- Tailored setup support from Cassotis Consulting
- Customizations possible

*. . . contact [sales@ampl.com](mailto:sales@ampl.com) for details*