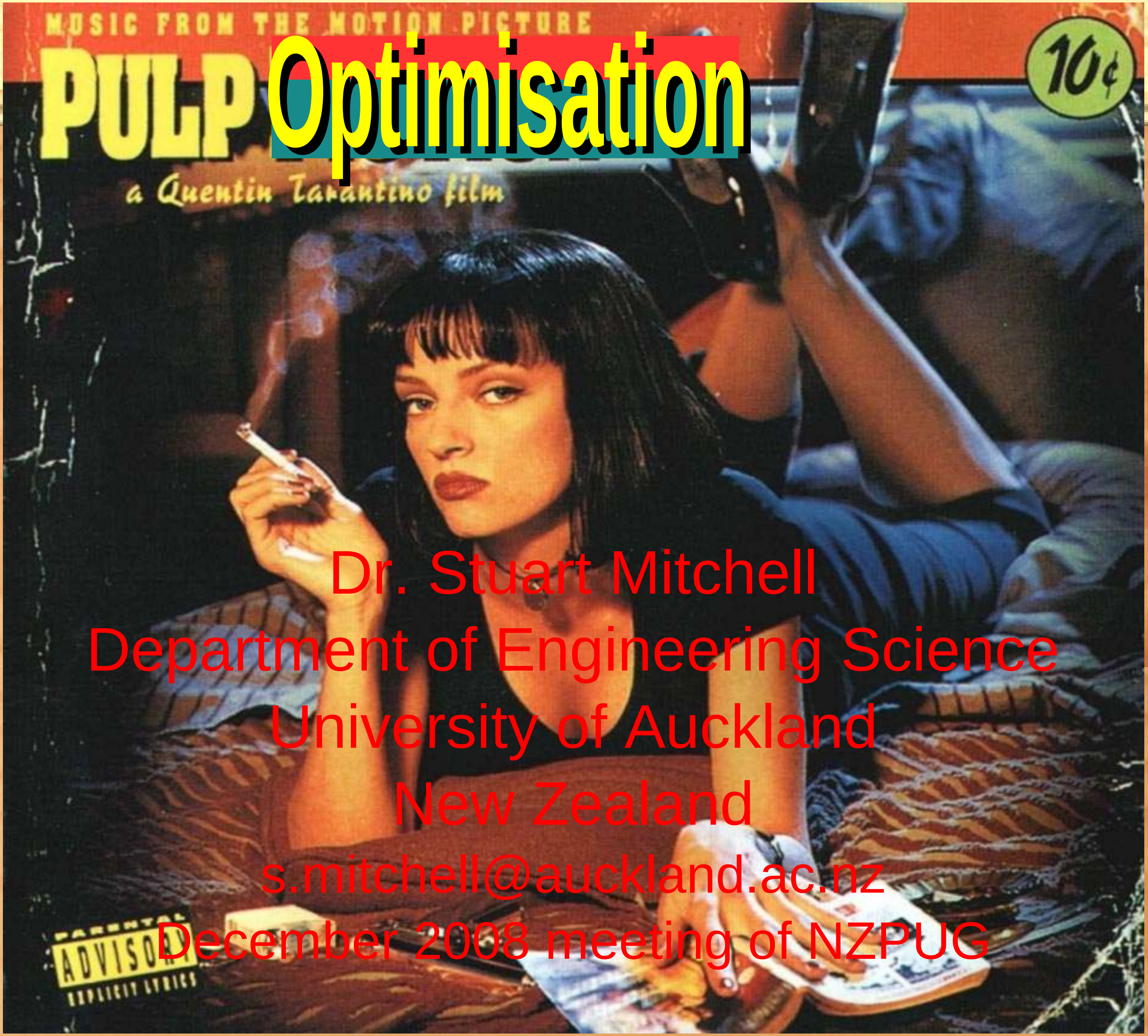
The
University
of Auckland

Optimisation



Dr. Stuart Mitchell
Department of Engineering Science
University of Auckland
New Zealand
s.mitchell@auckland.ac.nz
December 2008 meeting of NZPUG

Contents of presentation

What is Mathematical Programing

The Whiskas Problem

PuLP

Further PuLP examples

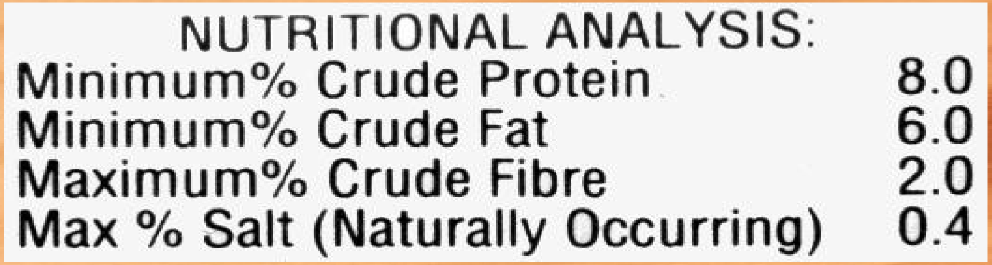
What is Mathematical programing

- A simple mathematically precise way of stating optimisation problems.
- Use mathematically rigorous ways to find a solution
- Examples of Optimisation Problems that can be solved with MP:
  - Shortest Path Problem
  - Scheduling Problems (Set partitioning)
  - Knapsack problems
  - Blending Problems

The Whiskas blending problem

- Taken from http://130.216.209.237/engsci392/pulp/ABlendingProblem
- Whiskas cat food want to produce their cat food products as cheaply as possible while ensuring they meet the stated nutritional analysis requirements shown on the cans.
- Thus they want to vary the quantities of each ingredient used (the main ingredients being chicken, beef, mutton, rice, wheat and gel) while still meeting their nutritional standards.

The Whiskas blending problem



NUTRITIONAL ANALYSIS:

Minimum% Crude Protein	8.0
Minimum% Crude Fat	6.0
Maximum% Crude Fibre	2.0
Max % Salt (Naturally Occurring)	0.4

	$/kg	Protein	Fat	Fibre	Salt
Chicken	$13	0.100	0.080	0.001	0.002
Beef	$8	0.200	0.100	0.005	0.005
Mutton	$10	0.150	0.110	0.003	0.007
Rice	$2	0.000	0.010	0.100	0.002
Wheat bran	$5	0.040	0.010	0.150	0.008
Gel	$1	-	-	-	-

Whiskas blending problem

- We wish to identify decision variables
- Assume we only have chicken and beef
- Let
  - x_c = the percentage of chicken meat
  - x_b = the percentage of beef used
- Then we wish to minimise $\$13x_c + \$8x_b$

Whiskas Blending Problem

- What about the nutritional requirements
- Subject to

  $x_c + x_b = 100$

  $0.100x_c + 0.200x_b >= 8.0$

  $0.080x_c + 0.100x_b >= 6.0$

  $0.001x_c + 0.005x_b <= 2.0$

  $0.002x_c + 0.005x_b <= 0.4$
- $x_c >= 0$, $x_b >= 0$

MP Model

$Let: I \in \{c, b, m, w, g\}$ the set of ingredients

x_i be the percentage of ingredient i in the cat food $i \in I$

C_i be the cost of ingredient i $i \in I$

P_i be the protien content of ingredient i $i \in I$

F_i be the fat content of ingredient i $i \in I$

Fb_i be the fibre content of ingredient i $i \in I$

S_i be the salt content of ingredient i $i \in I$

$$
\begin{aligned}
& min \sum_{i \in I} C_i x_i \\
& s.t. \\
& \sum_{i \in I} x_i = 100 \\
& \sum_{i \in I} F_i x_i \geq 8 \\
& \sum_{i \in I} Fb_i x_i \leq 2 \\
& \sum_{i \in I} S_i x_i \leq 0.4 \\
& x_i \geq 0 \; \forall i \in I
\end{aligned}
$$

Ok where is the python

- On google code you can find pulp-or http://code.google.com/p/pulp-or/
- This is a python module that allows the easy statement and solution of linear programing problems.
- Pulp leverages features of python and the open source optimisation libraries Coin-or

Whiskas model in python

```
"""
The Full Whiskas Model Python Formulation for the PuLP Modeller
Authors: Antony Phillips, Dr Stuart Mitchell  2007
"""
# Import PuLP modeler functions
from pulp import *
# Creates a list of the Ingredients
Ingredients = ['CHICKEN', 'BEEF', 'MUTTON', 'RICE', 'WHEAT', 'GEL']
# A dictionary of the costs of each of the Ingredients is created
costs = {'CHICKEN': 0.013,
         'BEEF': 0.008,
         'MUTTON': 0.010,
         'RICE': 0.002,
         'WHEAT': 0.005,
         'GEL': 0.001}
# A dictionary of the protein percent in each of the Ingredients is created
proteinPercent = {'CHICKEN': 0.100,
                  'BEEF': 0.200,
                  'MUTTON': 0.150,
                  'RICE': 0.000,
                  'WHEAT': 0.040,
                  'GEL': 0.000}
# A dictionary of the fat percent in each of the Ingredients is created
fatPercent = {'CHICKEN': 0.080,
              'BEEF': 0.100,
              'MUTTON': 0.110,
              'RICE': 0.010,
              'WHEAT': 0.010,
              'GEL': 0.000}
# A dictionary of the fibre percent in each of the Ingredients is created
fibrePercent = {'CHICKEN': 0.001,
                'BEEF': 0.005,
                'MUTTON': 0.003,
                'RICE': 0.100,
                'WHEAT': 0.150,
                'GEL': 0.000}
```

```
# Create the 'prob' variable to contain the problem data
prob = LpProblem("The Whiskas Problem", LpMinimize)

# A dictionary called 'Vars' is created to contain the referenced Variables
vars = LpVariable.dicts("Ingr",Ingredients,0)

# The objective function is added to 'prob' first
prob += lpSum([costs[i]*vars[i] for i in Ingredients]), "Total Cost of Ingredients per can"

# The five constraints are added to 'prob'
prob += lpSum([vars[i] for i in Ingredients]) == 100, "PercentagesSum"
prob += lpSum([proteinPercent[i] * vars[i] for i in Ingredients]) >= 8.0, "ProteinRequirement"
prob += lpSum([fatPercent[i] * vars[i] for i in Ingredients]) >= 6.0, "FatRequirement"
prob += lpSum([fibrePercent[i] * vars[i] for i in Ingredients]) <= 2.0, "FibreRequirement"
prob += lpSum([saltPercent[i] * vars[i] for i in Ingredients]) <= 0.4, "SaltRequirement"

# The problem data is written to an .lp file
prob.writeLP("WhiskasModel2.lp")

# The problem is solved using PuLP's choice of Solver
prob.solve()

# The status of the solution is printed to the screen
print "Status:", LpStatus[prob.status]

# Each of the variables is printed with it's resolved optimum value
for v in prob.variables():
    print v.name, "=", v.varValue

# The optimised objective function value is printed to the screen
print "Total Cost of Ingredients per can = ", value(prob.objective)
```

The open source future for OR

- Presently the Computational OR tools used, taught within this department, are closed source.
  - Excel /Storm
  - AMPL, GAMS
  - CPLEX, EXPRESS, ZIP
- Students can not afford commercial licences of this software
- Students cannot see how this software works.

The open source future for OR

- Outcomes for students
  - Ability to access free (no cost) software to implement their own solutions once they graduate
  - Ability to access free (open) source code to see how the algorithms are implemented.
    - Imagine the difference to 391??
  - The ability to improve the software they use.

PuLP

- PuLP is a python module that allows the easy expression of Mathematical Programs
- PuLP is built to interface with separate solvers
- PuLP is similar in style to:
  - AMPL
  - GAMS
  - OPL
  - LINGO
  - FLOPC++ etc.

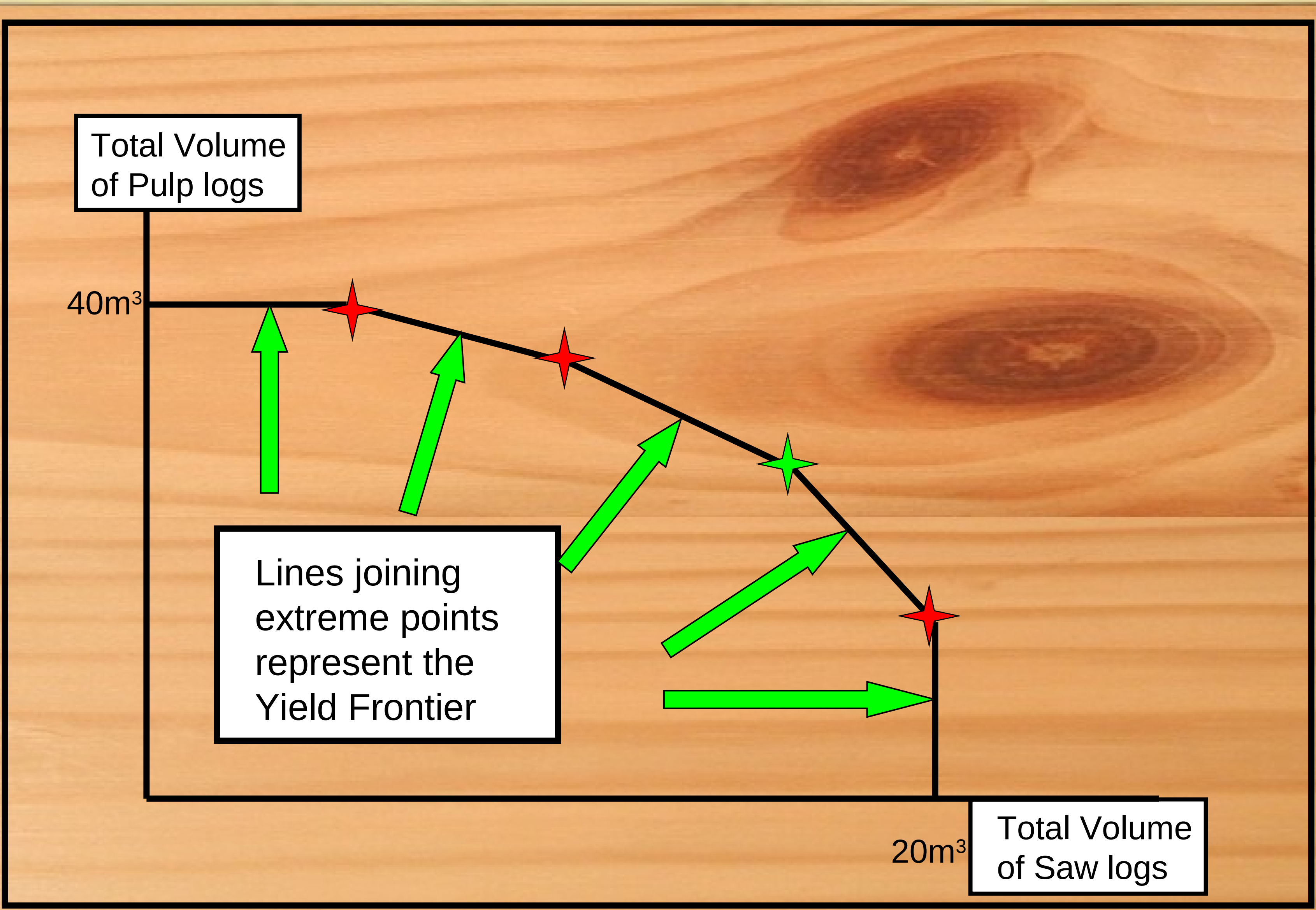
PuLP

- Why Python?
  - Core Python syntax leads to the concise statement of MP's
  - Python is a scripting language so no compilation is needed and the code is platform independent
  - Python interfaces easily with external solvers that do the heavy lifting
  - Python comes with 'batteries included'
    - The Python standard library is huge

PuLP

- Written initially by J. S. Roy
- Now maintained by S. A. Mitchell
- It is available at http://pulp-or.google-code.com
- Now available for Windows and Linux

Generating a Yield Frontier



Lines joining extreme points represent the Yield Frontier

Generating a Yield Frontier

- Using pulp we formulate the bucking problem (with a single objective) as a set packing problem by log section.

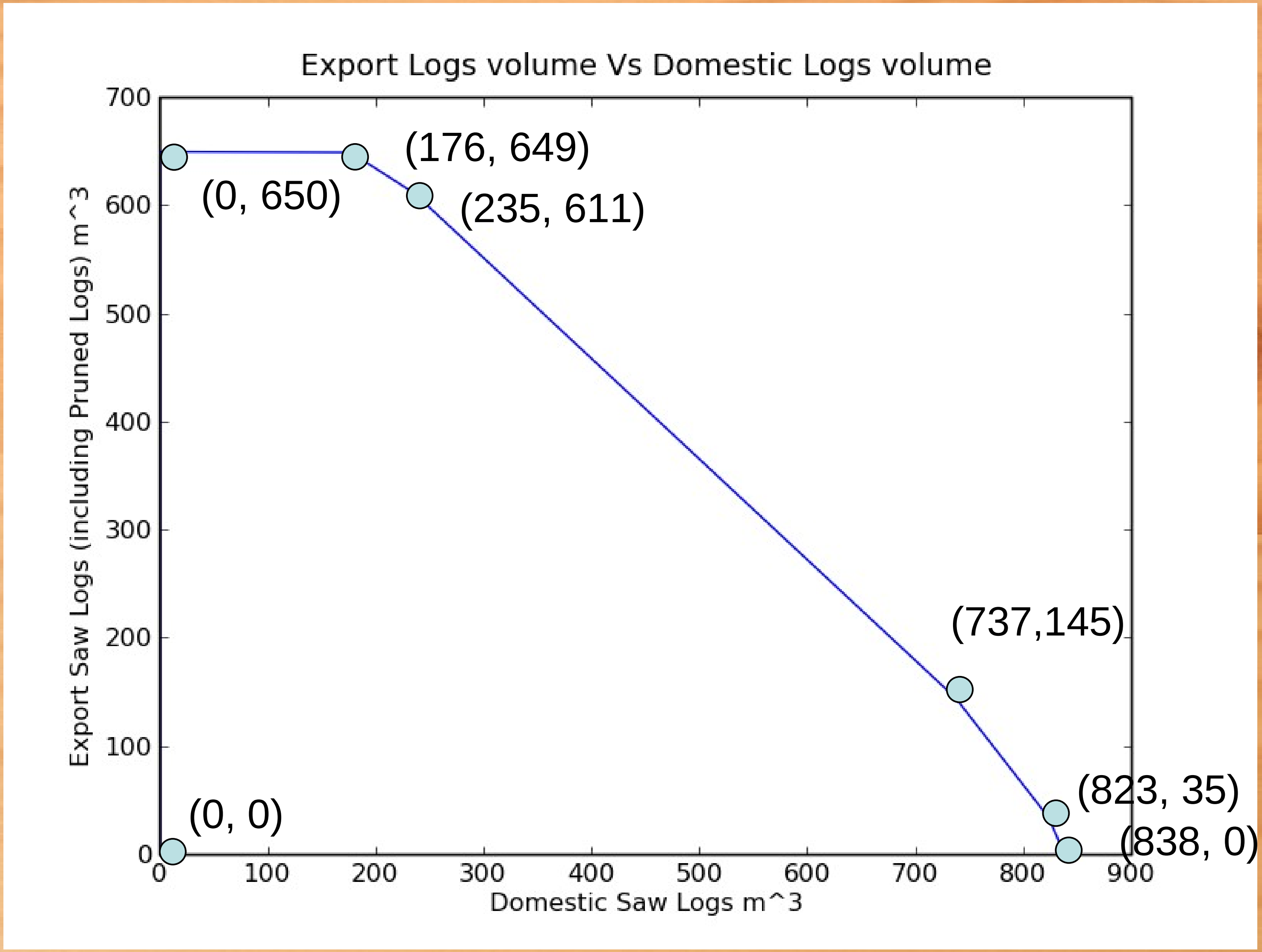
```
lp = LpProblem("Bucking Model", LpMaximize)
#set up the logvolume variables
logvol=LpVariable.dicts("LogVolume(\"%s\")",logtypes,0)
#objective
lp+=lpSum([l.price * logvol[l] for l in logtypes]), "Revenue“
#setup the arc variables
x=LpVariable.dict("x(%s)",f_logs,0,1,LpInteger)
#set up a section set partitioning problem
count = 0
for s in stems:
        slogs = fs_logs[s]
        for i,sec in enumerate(s.sections):
            lp +=( lpSum((x[log] for log in slogs
                         if log.startl <= sec.start
                         if log.endl > sec.start)) <= 1
                         , "Stem_Section(\"%s\",%i)" % (str(s),i))
            count += 1
#add the constraints that link the log volumes
for lt in logtypes:
        lp +=( lpSum((log.volume*x[log]
                     for log in fl_logs[lt])) - logvol[lt] == 0
                     , "Logtype_volume(\"%s\")" % str(lt))
```

Generating a Yield Frontier Using Pulp

- We then iteratively solve the problem to find all extreme supported solutions on the Yield Frontier
- Equivalent to projecting the problem into the log volume space
- I added a module to PuLP that implements projection using Iterative Hull Methods (Lassez, Lassez 1992)

```
>>> pprob, ppoints = polytope.project(lp, totalvars)
```

Find Yield Frontier for the Dataset



Find Yield Frontier for a Single Stem

```
\* Total projected *\
Minimize
OBJ: __dummy
Subject To
_C1: DomSaw + 1.11154598826 ex <= 2669.27592955
_C2: DomSaw + 1.34653465347 ex <= 3118.57425743
_C3: 1.00863930886 DomSaw + ex <= 2522.60691145
Bounds
__dummy = 0
End
```

Travelling tournament problem with PuLP

- This problem models the allocation of teams to Home and Away games in a tournament
- A full problem description and datasets are found at Michael Trick's page
- http://mat.tepper.cmu.edu/TOURN/

Travelling tournament problem with PuLP

- At IFORS 2008 M. Trick presented an approach to finding lower bounds to this problem using combinatorial benders cuts
- That evening I implemented his algorithm using PuLP
- Along the way I also added Powerset, Combination and Permutation operators to PuLP

-
  ```
  lp = LpProblem("Travelling tournement Master", LpMinimize)
  #create variables
  triplist = [Trip(t1,p) for t1 in teams
                  for p in
                  allpermutations([t for t in
                  teams if t !=t1] ,k)
                  if p[0] <= p[-1]]
  tripvars = LpVariable.dict("mastervar ",triplist,0,1,LpInteger)
  #objective
  lp += lpSum([t.cost()*tripvars[t]
                        for t in triplist])
  #construct constraints to ensure that all teams visit each other
      for t1 in teams:
          for t2 in teams:
              if t1 != t2:
                  lp += lpSum([tripvars[t] for t in triplist
                                        if t.team == t1
                                        if t2 in a.awayteams]) == 1, \
                                       "Team_%s_Visits_%s"%(t1,t2)
  ```

Further examples

- The 392 course has been converted from AMPL to PuLP http://130.216.209.237/engsci392/pulp/OptimisationWithPuLP
- There you can see a number of different ways to construct problems
- Note that new language features can be added very easily only needing approval from the BDFL