SAND2011-2647C

Pyomo: Modeling and Solving Mathematical Programs in Python

William E. Hart

Sandia National Laboratories

wehart@sandia.gov

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Why Math Modeling?

Goals:

- Provide a natural syntax to describe mathematical models
- Formulate large models with a concise syntax
- Separate modeling and data declarations
- Enable data import and export in commonly used formats

Impact:

- Robustly model large constraint matrices (e.g. for MILPs)
- Integrated support of automatic differentiation for complex nonlinear models

Examples: AMPL, GAMS, OptimJ, AIMMS, FlopCPP, PuLP, …

Pyomo Features

Open Source
- Transparency and reliability
- Customizable capability
- Flexible licensing

Flexible Modeling Language
- Leverages a full-featured, modern scripting language
- Extensible library of Python modeling objects

Portability
- Linux, MS Windows, Mac OS

Solver Integration
- Tight integration: solvers linked into modeling language
- Loose integration: solver launched separately

Flexible Modeling Environment
- Support for LP, MILP and NLP models
- Symbolic/Concrete representations of objectives and constraints
- Construct models from external data sources
  - Databases, spreadsheets, Pyomo data files, CSV files, etc.
- Modeling extension packages
  - Generalized disjunctive programming, stochastic programming

Why Python?

Open Source License

Features

- A clean syntax, a rich set of data types, support for object oriented programming, namespaces, exceptions, dynamic loading, etc.

Support and Stability

- Highly stable and well-supported

Documentation

- Extensive online documentation and several excellent books

Standard Library

- Includes a large number of useful modules.

Extendability and Customization

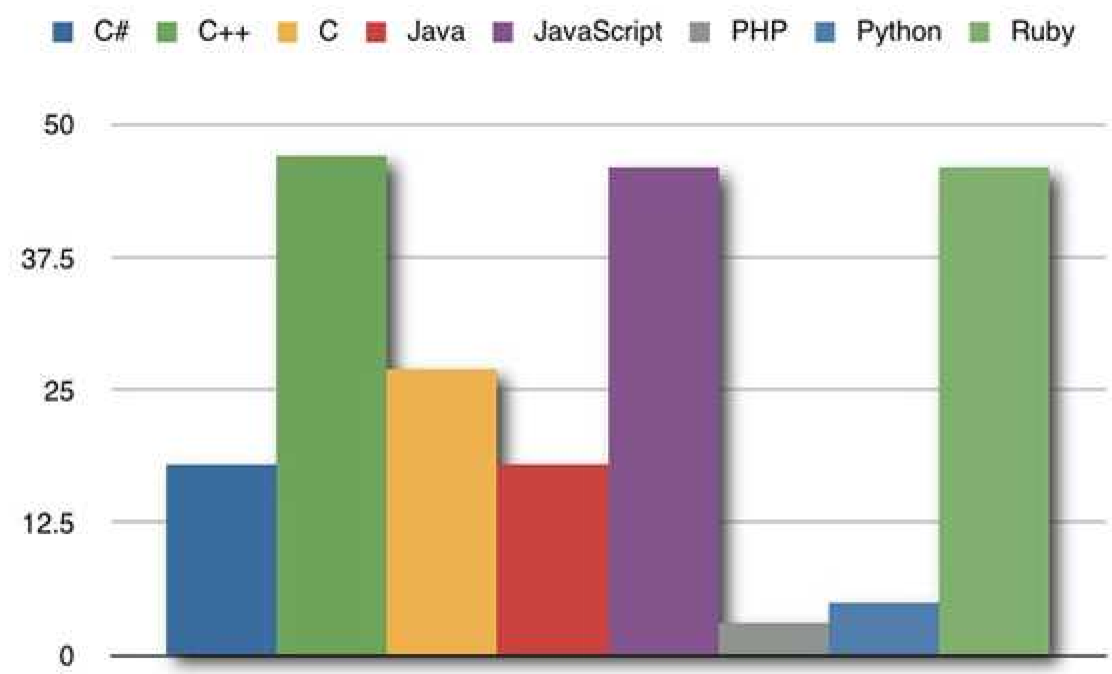
- Simple model for loading Python code developed by a user
- Can easily integrate libraries that optimize compute kernels
- Python can dynamically integrate libraries

Portability

- Widely available on many platforms

Why Python? More work and few expletives!

An analysis of cuss words in Git Hub software...



See http://www.andrewvos.com/ for further details...

Example: A Concrete Knapsack Model

```
from coopr.pyomo import *

v = {'hammer':8, 'wrench':3, 'screwdriver':6, 'towel':11}
w = {'hammer':5, 'wrench':7, 'screwdriver':4, 'towel':3}

limit = 14

model = ConcreteModel()

model.ITEMS = Set(initialize=v.keys())

model.x = Var(model.ITEMS, within=Binary)

model.value = Objective(expr=sum(v[i]*model.x[i] for i in
  model.ITEMS), sense=maximize)

model.weight = Constraint(expr=sum(w[i]*model.x[i] for i in
  model.ITEMS) <= limit)
```

Using the Pyomo Command-line Script

Pyomo's command-line script executes a common workflow

Executing ...

```
$ pyomo --solver=glpk knapsack-concrete.py
```

... generates the following:

```
- Setting up Pyomo environment                                     [    0.00]
- Applying Pyomo preprocessing actions                             [    0.00]
- Creating model                                                   [    0.01]
- Applying solver                                                  [    0.03]
- Processing results                                               [    0.30]
    Number of solutions: 1
    Solution Information
      Gap: 0.0
      Status: optimal
      Function Value: 25
    Solver results file: results.yml
- Applying Pyomo postprocessing actions                            [    0.30]
- Pyomo Finished                                                   [    0.30]
```

Example: An Abstract Knapsack Model

```
from coopr.pyomo import *

model = AbstractModel()

model.ITEMS = Set()

model.v = Param(model.ITEMS, within=PositiveReals)

model.w = Param(model.ITEMS, within=PositiveReals)

model.limit = Param(within=PositiveReals)

model.x = Var(model.ITEMS, within=Binary)

def value_rule(model):
    return sum(model.v[i]*model.x[i] for i in model.ITEMS)
model.value = Objective(sense=maximize)

def weight_rule(model):
    return sum(model.w[i]*model.x[i] for i in model.ITEMS) <=
  model.limit
model.weight = Constraint()
```

Optimizing Abstract Models

Pyomo supports the integration of data using a Pyomo data commands

- These are specified in a separate file
- Data commands are a mini-language that is closely resembles AMPL's data commands

Example: knapsack.dat

```
set ITEMS := hammer wrench screwdriver towel ;

param : v w :=
    hammer       8 5
    wrench       3 7
    screwdriver  6 4
    towel       11 3;

param limit := 14;
```

```
$ pyomo --solver=glpk knapsack-abstract.py knapsack.dat
```

Data Interfaces

Note: the ‘set’ and ‘param’ data commands allow for compatibility with AMPL models

The ‘import’ command that can be used to load data from a variety of external data sources:

- databases, excel spreadsheets, CSV files, tabular data, etc.

Example: diet1.db.dat

```
import "Driver={Microsoft Access Driver (*.mdb)};DBQ=diet1.mdb"
   using=pyodbc
   query="SELECT FOOD,cost,f_min,f_max FROM Food":
      [FOOD] cost f_min f_max;
```

Example: Nonlinear Models

Pyomo now supports general nonlinear expressions for objectives and constraints

A generic ASL solver can be used to apply any solver that uses the AMPL Solver Library

```
from coopr.pyomo import *

model = ConcreteModel()

model.x1 = Var()
model.x2 = Var(bounds=(-1,1))
model.x3 = Var(bounds=(1,2))

def obj_rule(m):
    return m.x1**2 + (m.x2*m.x3)**4 +
          m.x1*m.x3 +
          m.x2*sin(m.x1+m.x3) + m.x2
model.obj = Objective(sense=minimize)
```

Example - solve this problem with ipopt from a Unix command line:

```
$ pyomo --solver=asl:ipopt eg1.py
```

Comparison with Other Python Modeling Tools

- Pyomo
  - Supports concrete/abstract modeling for LP/MILP/NLP models
  - Separate model objects
  - Distributed package architecture is easily extensible, but complex
- PuLP
  - Supports concrete modeling for LP/MILP models
  - Separate model objects
  - Single Python package that is easy to install
- APLEpy
  - Supports concrete modeling for LP/MILP models
  - Single global model object
  - Single Python package that is easy to install
- PyMathProg, pyglpk, cplex, gurobi
  - Python interfaces for specific solver tools

Some Noteworthy Limitations of Pyomo

- Pyomo only works with Python 2.6 and 2.7
- Pyomo object/constraint declarations are more verbose than AMLs
  - Typically requires the use of a temporary function
- Pyomo does not include preprocessing of LP/MILP instances
- Instance generation can be much slower than commercial tools
  - But we're catching up...!
- Pyomo only has a simple GUI driver
- Pyomo installation requires a variety of Python packages

Coopr: A COmmon Optimization Python Repository

Coopr integrates Python packages related to modeling and optimization

- **coopr.age** A GUI for formulating and solving models
- **coopr.gdp** Extension package for disjunctive programming
- **coopr.opt** Generic interfaces for optimization solvers
- **coopr.pyomo** A Pythonic optimization modeling tool
- **coopr.pysp** Pyomo stochastic programming extensions for Pyomo
- **coopr.neos** Extension package for the NEOS solvers

Pyomo is a keystone project within Coopr

- Pyomo is designed to facilitate extensions
- Many Coopr projects extend Pyomo's modeling capabilities

Coopr Solvers

- **asl**
  Shell interface to a generic optimizer that uses the AMPL Solver Library to interface with a math programming model
- **cbc**
  Shell interface to the CLP/CBC LP/MIP solver
- **cplex**
  Shell interface to the CPLEX LP/MIP solver
- **cplexdirect**
  Direct Python interface to the CPLEX LP/MIP solver
- **glpk**
  Shell interface to the GNU Linear Programming Kit
- **gurobi**
  Shell interface to the GUROBI LP/MIP solver
- **pico**
  Shell interface to the PICO MIP solver

Coopr Releases

Coopr 2.4

- Release on 10/29/2010 (600+ code commits)
- Lots of modeling enhancements: concrete models, nonlinear modeling, SOS constraints, piece-wise linear components
- coopr.age GUI
- Data interfaces for relational databases

Coopr 2.5

- Release in early March, 2011
- Significant improvements in memory/speed
- Modeling disjunctive programs
- Simplified command-line interface
- MS Windows installer
- Bug fixes!?!

Coopr Developers

- **Sandia National Laboratories**
  - William Hart [Coopr & coopr.pyomo project lead]
  - Jean-Paul Watson [coopr.pysp project lead]
  - John Siirola [coopr.gdp project lead]
  - Tom Brounstein
- **University of California, Davis**
  - Prof. David L. Woodruff [coopr.pysp co-developer]
  - Prof. Yueyue Fan
- **Texas A&M University**
  - Prof. Carl D. Laird [coopr.age project lead]
  - Daniel Word
  - James Young
  - Gabe Hackebeil
- **William & Mary**
  - Patrick Steele
- **North Carolina State**
  - Kevin Hunter

Getting Started

Sandia Coopr wiki

https://software.sandia.gov/trac/coopr/

Installation Documentation

https://software.sandia.gov/trac/coopr/wiki/GettingStarted

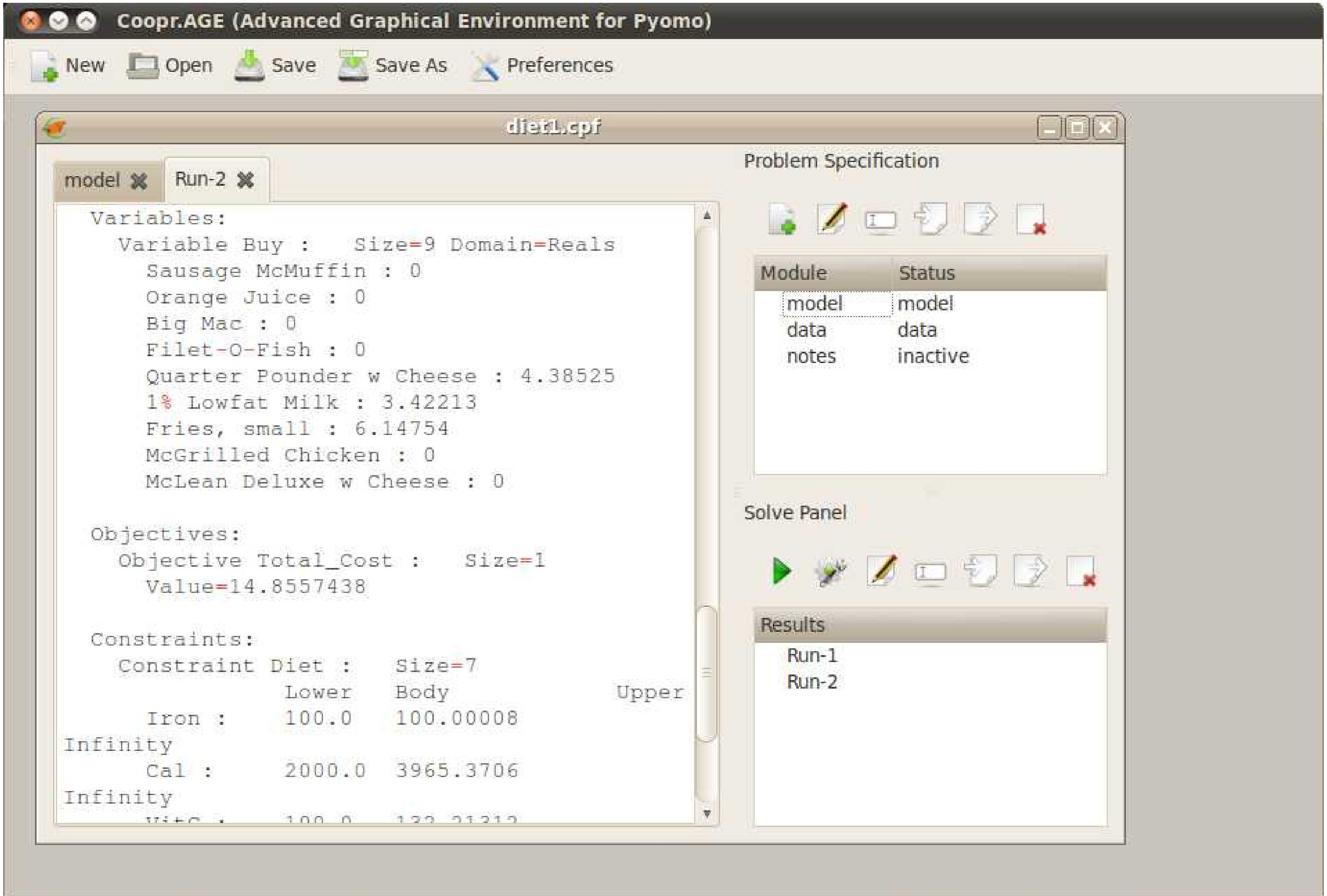
Coopr Forum: a mailing list for help and announcements

http://groups.google.com/group/coopr-forum

CoinBazaar: a COIN-OR project that supports Coopr extension packages

https://projects.coin-or.org/CoinBazaar

The coopr.age GUI



PyPI Distribution

Note: Coopr is comprised of a set of distinct Python packages

- This can complicate installation...
- Coopr can be downloaded from PyPI to simplify installation
  - The following examples work on Linux

If you have administrative privileges, then you can install Coopr in your Python installation as a site package:

1. Download and install the setuptools package
   - wget http://peak.telecommunity.com/dist/ez_setup.py
   - python ez_setup.py

2. Run easy_install to install Coopr:
   - easy_install Coopr

PyPI Distribution

If you do *not* have administrative privileges, then use the coopr_install script to create a virtual Python installation:

1. Download the coopr_install script
   – wget http://goo.gl/HVCVc

2. Create the virtual python installation in a specified directory
   – python coopr_install coopr

The coopr/bin directory contains a python command that has Coopr installed as a site package!