Stochastic modeling software in the open-source COIN-OR repository

Alan King

IBM Research
kingaj@us.ibm.com

INFORMS
Seattle, Nov 2007

Outline

Stochastic Modeling in COIN-OR

There are two projects that have stochastic in their goals:

- SMI - Stochastic Modeling Interface
  - Alan King, IBM Research
- FlopC++ - Formulation of Linear Optimization Problems in C++
  - Tim Hultberg, EUMETSAT

Some other projects that could be/are helpful

- DFO - derivative-free optimization
- CSDP - semi-definite programming
- OSI - open solver interface

COIN-OR Project SMI

Coin-SMI

- An interface for problems in which uncertainty and optimization appear together.
- Manages complex programming issues: implement multi-stage probability distributions, interact with solvers, etc.
- Many models *could* be supported: recourse programming, chance constrained programming, stochastic control and dynamic programming, robust optimization, queues, etc

Current Release as of 08/2007

The current release implements a multi-stage stochastic linear program with recourse class *SmiScnModel*:

- SMPS file reader.
- Direct input methods for scenarios and discrete distributions.
- Supports *all* OSI-compatible solvers.
- Prepares OSI object with deterministic equivalent LP.
- Traverses scenario tree and retrieves solution data by stage and scenario.
- Fully native COIN implementation.

Would be easy to add integer variables and quadratic objectives.

What challenges does SmiScnModel meet?

Automates problem generation, manages solver interactions, and produces solution reports for multi-stage stochastic linear programs.

- Accept user definitions of SLP data:
  - LP data,
  - Time stages,
  - Distributions.
- Process distribution information into scenario tree.
- Provide deterministic equivalent to OSI optimizers.
- Manage optimization processes (eg SAA).
- Report solution data back to user environment by stage and scenario.

Examples of usage in the Smi distribution

`COIN/Examples/Stoch/stoch.cpp` demonstrates several use cases.

- `SmpsIO("../../Mps/Stochastic/wat_10_C_32");` solves an SMPS example from Dempster et al: Watson Asset Liability problems.
- `ModelScenario();` scenario generation for Dantzig-Ferguson's Aircraft Allocation problem.
- `ModelDiscrete();` discrete distribution version of Aircraft Allocation problem.

Major pain points for users

Translating impact of random event into LP terms.

- Random events may impact multiple entries in LP – has to be specified by LP row index and column index.
- Situation gets even more complicated when implementing Galerkin-type solution methods.

Specification of distributions.

- Scenarios might just come from a "black-box" simulator, about which you know nothing.

Serial interaction with solver.

- You don't want all that scenario data + solution on your laptop.
- Have to interact with SMI - to generate scenarios, multiple times during solution, and multiple times to query solutions.

So we are going to take a look at FlopC++.

COIN-OR Project FlopC++

COIN-FLOPC++

- Specify linear optimization models in an style similar to Algebraic Modeling Language.
- Strength lies in the fact that the modeling facilities are combined with a powerful general purpose programming language (C++).
- Capability to implement efficient algorithms (using linear optimization for subproblems) and to integrate optimization models in user applications.
- Several examples in the distro are stochastic: aircraft, stochBenders, stAMPL.

A first take on a stochastic LP format

Financial Planning model of Lopez and Fourer

```
MP_set I(numI);
MP_stage T(numStages);
MP_stochastic_data R(T,I);
MP_variable Buy(T,I), Short, Over;
MP_constraint Invest, Roll(T), Goal;
Invest()   = sum(I, Buy(0,I))
                  == initial_wealth;
Roll(T+1) = sum(I, Buy(T,I) * R(T,I))
                  == sum(I, Buy(T+1,I));
Goal()     = sum(I, Buy(T.last(),I) * R(T.last(),))
                  == goal - Short() + Over();

maximize( Over() - 4*Short() );
```

Points to notice

- Notice the elegant use of operator overloading

  `operator == (left, right)`

  Similarly for >= and <= and so forth. Brilliant!

- Also take note: the flopc++ model captures state of member objects at the time they are added.
- Subsequent changes to member objects are not noticed.

Connecting FLOPC++ to Stochastic Programming

Easy to implement specialized stochastic data structures

- `MP_stage` labels data/rows/columns by stage.
- `MP_stochastic_data` Discrete distributions can be passed directly to SMI – as in Aircraft problem.

By introducing "update" capability into FlopC++ can implement methods like solving with expected values:

```
MP_stochastic_data R(T,I);
/* ... */
R.generateExpectedValues();
mp_model.solve();
```

... or attaching a simulation object and building a simulated model:

```
R.attachSimulator(simulator);
mp_model.generateSimulation();
```

Idea is that `MP_stochastic_data` manipulations cause updates to constraints, variables, and model.

Some philosophical concerns

Just like SMPS, clearly important aspects of the model are implicit!

- `Buy` variable should be non-anticipative.
- `Roll` constraints should be non-anticipative
- `Goal` constraint should be measureable `T.last()`.
- Objective should maximize Expected Value

In some ways this implicitness is also an attractive part of the elegance of SMPS definitions.

Doing it differently – beyond SMPS...maybe

- `maximize E( Over() - 4*Short())`
- This differentiates from maximizing for each sample point – also a reasonable interpretation of a problem that contains stochastic data.
- But again it is implicit that `Over()` and `Short()` are `T.last()` measurable random variables. Perhaps they should be specified with `T.last()` dependence.
- Should we introduce $E(\cdot|T)$ – conditional expectation relative to the filtration?
- How about a *CVaR* constraint in a multi-stage investment problem...?

Summary

- Introduced COIN-OR an opensource repository and community.
- Discussed Stochastic Modeling Interface (SMI) for modeling stochastic programming problems.
- Introduced FlopC++ stochastic modeling initiative.
- Discussed some prospective developments in stochastic modeling with SMI and FlopC++.
- Goal of SMI-FlopC++ is to provide a programming environment to develop coding patterns for SP modeling and solution – that someday may find their way into commercial modeling environments.