U.S. & Mexico Workshop on Optimization and its Applications

Huatulco, Mexico, 8–12 January 2018

Toward Conveniently Handling Bi-Level Optimization Problems

David M. Gay

AMPL Optimization, Inc.

Albuquerque, New Mexico, U.S.A.

dmg@ampl.com

http://www.ampl.com

Background: AMPL, a language for mathematical programming, e.g.,

$$\text{minimize } f(x)$$
$$\text{s.t. } \ell \leq c(x) \leq u,$$

with $x \in \mathbb{R}^n$ and $c : \mathbb{R}^n \to \mathbb{R}^m$ given algebraically and some x_i discrete.

Motivation for bilevel optimization

Sometimes a decision affects other parties who can take recourse.

Modeling their recourse actions as *inner* optimization problems may be appropriate: the decision maker has an outer optimization problem with inner optimization problems as constraints.

Disclaimer

Economists, game theorists, and complementarity researchers have looked at nested problems for many years. The present goal is to examine such problems from an AMPL perspective, with an eye to using automatic differentiation to help formulate and solve them.

Toy inner optimization problem (inner.x)

```
# simple "inner" problem for toy bilevel example

param c default 0; # to be a variable in the bilevel problem
var x := 1;
var x1 := 3;
var x2 := 0;
s.t. circle: (x1 - 4)^2 + x2^2 == 1;
# distance to the parabola y = x^2 + c
minimize dist: (x - x1)^2 + (x^2 + c - x2)^2;
```

Toy bilevel example (bilev.x)

```
# bilevel variant with modified outer objective and
# explicit first-order nec. cond's for inner obj.
var c := 1;
var x := 1;
var x1 := 3;
var x2 := 0;
var dist = (x - x1)^2 + (x^2 + c - x2)^2;
s.t. circle: (x1 - 4)^2 + x2^2 == 1;
var lambda := -1.6;
minimize bilev: c^2 + dist;
s.t. nec1: x - x1 + lambda*(x1-4) == 0;
s.t. nec2: x^2 - c - x2 + lambda*x2 == 0;
```

Solving inner.x

```
ampl: model inner.x; solve;
MINOS 5.51: optimal solution found.
12 iterations, objective 4.584878775
Nonlin evals: obj = 32, grad = 31, constrs = 32, Jac = 31.
ampl: display _varname, _var;
: _varname      _var       :=
1   x          1.12817
2   x1         3.08576
3   x2         0.405184
;
```

Solving bilev.x

```
ampl: reset; model bilev.x; solve;
MINOS 5.51: optimal solution found.
17 iterations, objective 4.246188161
Nonlin evals: obj = 44, grad = 43, constrs = 44, Jac = 43.
ampl: display _varname, _var, dist;
: _varname      _var         :=
1    c          -0.409548
2    x           1.24962
3    x1          3.18718
4    x2          0.582515
5    dist        4.07846
6    lambda     -2.38376
;

dist = 4.07846
```

Discussion

Necessary conditions for problems with inequality constraints involve *complementarity* constraints, e.g.,

```
s.t.  c:  lambda >= 0 complements f(x) >= 0;
```

Manually stating necessary conditions is error prone, so AMPL should automatically provide these conditions.

AMPL's *problem* facility

An AMPL *problem* declaration lists variables, constraints and objectives for a named problem. Other variables are held fixed, and other constraints are ignored. A named problem can also have its own *environment*.

Named problem example (cutting stock)

```
param nPAT integer >= 0;  # number of patterns
set PATTERNS = 1..nPAT;   # set of patterns
var Cut {PATTERNS} integer >= 0;
minimize Number: sum {j in PATTERNS} Cut[j];
s.t. Fill {i in WIDTHS}: ...;
var Use {WIDTHS} integer >= 0;
minimize Reduced_Cost: ...;
s.t. W_Limit: ...;
problem Cutting_Opt: Cut, Number, Fill;
option relax_integrality 1;
problem Pattern_Gen: Use, Reduced_Cost, W_Limit;
option relax_integrality 0;
```

Named problem use example

```
repeat {
    solve Cutting_Opt;
    let {i in WIDTHS} price[i] := Fill[i].dual;

    solve Pattern_Gen;
    if Reduced_Cost >= -0.00001 then break;
    let nPAT := nPAT + 1;
    let {i in WIDTHS} nbr[i,nPAT] := Use[i];
    };
```

See
https://ampl.com/BOOK/CHAPTERS/17-solvers.pdf

Extending AMPL's *problem* facility

Proposal: Allow problem declarations to list inner named problems. AMPL would supply first-order necessary conditions for inner problems as constraints. Environments of inner problems would be ignored. Variables of inner problems would also be overall problem variables.

First-order necessary conditions for an inner problem

Lagrangian for

$$\text{minimize } f(x) \text{ s.t. } c(x) \geq 0$$

is

$$\psi(x, \lambda) = f(x) + \lambda c(x).$$

First-order necessary conditions:

$\nabla_x \psi(x, \lambda) = 0$ with $c(x) \geq 0 \perp \lambda \geq 0$.

Implied constraints for an inner problem

Plan: augment constraints a solver sees with first-order necessary conditions for inner problems. These conditions just involve partial derivatives with respect to the inner problem's variables. Such gradients are readily computed by reverse AD (Automatic Differentiation).

Chain rule: basis for automatic differentiation (AD)

Suppose for scalar x that

$$\phi(x) = f(y_1(x), y_2(x), ..., y_k(x)).$$

The chain rule gives

$$\frac{\partial \phi}{\partial x} = \sum_{i=1}^{k} \frac{\partial f}{\partial y_i} \frac{\partial y_i}{\partial x} = \sum_{i=1}^{k} \frac{\partial \phi}{\partial y_i} \frac{\partial y_i}{\partial x}.$$

In general, once we know the *adjoint* $\frac{\partial \phi}{\partial y}$ of an intermediate variable y, we can add its contribution $\frac{\partial \phi}{\partial y} \frac{\partial y}{\partial x}$ to the adjoint $\frac{\partial \phi}{\partial x}$ of each variable x on which y directly depends.

AD in the AMPL/solver interface library

Paper available from `http://ampl.com`: *Revisiting Expression Representations for Nonlinear AMPL Models* is about AD in the AMPL/solver interface library (ASL). DMG talk at the 2016 *U.S. and Mexico Workshop* in Merida was a preliminary version of this paper.

Jacobians for inner problems

Some nonlinear solvers (e.g., *minos* and *snopt*) only want to be given function and gradient values. For such solvers, gradients of implied constraints (i.e., Jacobian rows) amount to Hessians and are readily supplied by existing ASL facilities.

Computing Hessians

Other solvers (e.g., *conopt*, *ipopt*, *knitro*, *loqo*) want explicit Hessians or Hessian-vector products as well as function and gradient values. The ASL approach: compute $v^T \nabla^2 f(x)$ by considering $\phi(\tau) = f(x + \tau v)$; compute $\phi'(\tau)$ by forward AD and apply reverse AD to $\phi'(\tau)$, giving $(\nabla^2 f(x))v$.

More on computing Hessians

Equivalent way to regard $(\nabla^2 f(x))v$: apply reverse AD to $v^T \nabla f(x)$, giving its gradient.

When explicit $\nabla^2 f(x)$ is needed, ASL computes it one row at a time via Hessian-vector products.

Hessians for inner problems

Plan for inner-problem Hessians: consider

$$\phi(x, \tau, \sigma) = f(x + \tau v + \sigma w);$$

compute $\frac{\nabla^2 f}{\nabla \tau \nabla \sigma} = w^T \nabla^2 f(x) v$ by forward AD and apply reverse AD to obtain a row of the desired Hessian.

Hessians = challenge for inner problems

Current ASL Hessian-vector and full Hessian computations are tuned to outer problems. Generalizing to inner problems requires extensions to the "`.nl`" format and some ASL routines and an option to allow or exclude inner problems that determine some of the same variables.

Multi-level problems

When an inner problem itself is a bilevel problem, we have a tri-level problem. In general, we could have several levels, with the necessary conditions for an inner problem appearing as complementarity constraints to the containing problem.

Solving

Bi- and multi-level problems in general can be nonconvex, possibly difficult global optimization problems.

Some current solvers...

Solver	complem.	optim.	global
baron	No	Yes	Yes
knitro	Yes	Yes	No
path	Yes	No	No

We need solvers with three Yes's.

Partially separable structure

Some functions are *partially separable*:

$$f(x) = \sum_{i=1}^{q} \theta_i(f_i U_i x))$$

where θ_i is unary. An expression-graph walk finds this structure or more detailed "group partial separability", and using it can save time.

More on AMPL and AD therewith

The AMPL web site

`http://ampl.com`

has more on AMPL, including pointers to papers on AD with AMPL and on the AMPL/solver interface library (ASL).

For more on AD in general, see

`http://www.autodiff.org`