Experience with CGL in the PICO Mixed-Integer Programming Solver

Cynthia A. Phillips, Sandia National Laboratories

Joint work with

Jonathan Eckstein, Rutgers

William Hart, Sandia

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.

Parallel Integer and Combinatorial Optimizer (PICO)

Mixed-Integer programming solver built on top of PEBBL's general branch-and-bound framework

- Historical (and continuing) raison d'être: massively parallel (scalable)
  - Distributed memory (MPI), C++
- More recent interest in improved serial performance

- Portable, flexible
  - Serial, small LAN, Cplant, ASCI Red, Red Storm
- Allows exploitation of problem-specific knowledge/structure
- Open Source release
  - Always support a free LP solver

PICO uses Pieces of COIN-OR for Branch+Cut

- Osi - Open Solver Interface
  - PicoLPInterface class derived from this
- Cgl - Cut Generation Library
  - PICO cut finder wrapper class to use Cgl generators
- Clp - COIN LP Solver
  - Current PICO default

Local differences with official version

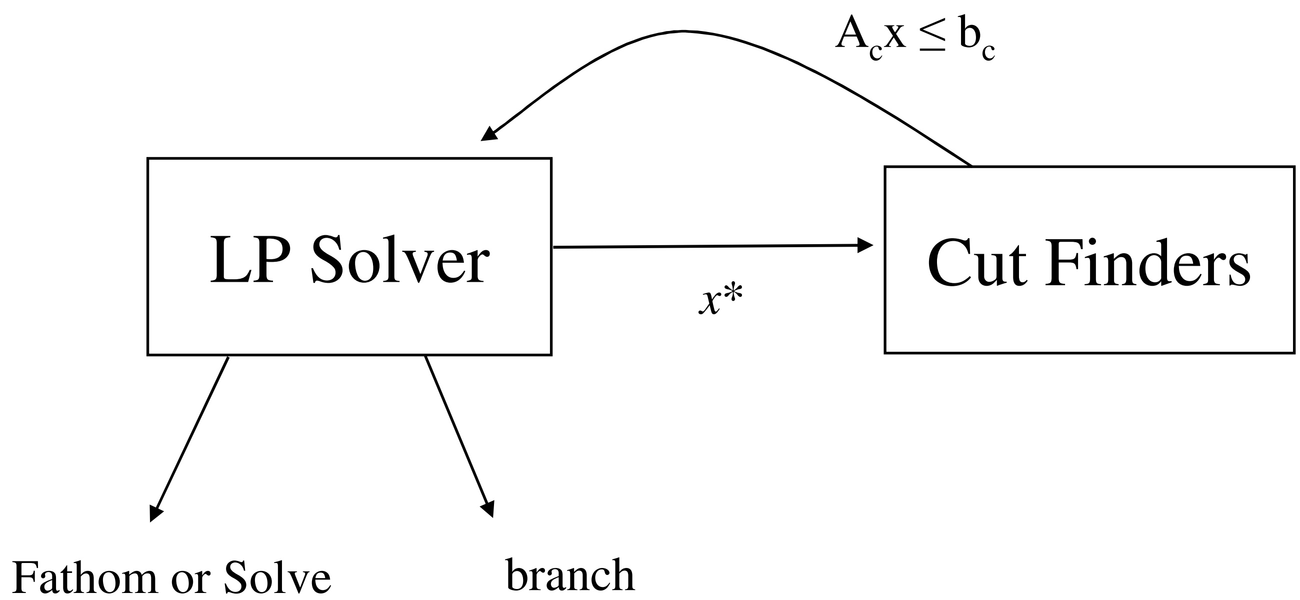
- Portability
- Bug fixes not yet incorporated (frequently component interaction)
- Some changes needed to compile derived classes
  - Some changes to data ownership, protection levels

COIN in ACRO

We maintain version of COIN pieces we use in Acro cvs repository

- In third-party packages section
- Single checkout of all pieces (PICO, PEBBL, utilib, COIN)
- ACRO daily QA
  - Daily checkout and build on many different platforms
    - Linux, solaris, Mac OS X, cygwin, Irix, SGI, etc
  - Daily tests
  - Daily tracking of changes in COIN
  - Daily summary emails to developers

Generic Branch and Cut of a MIP Subproblem



- In PICO, searching Pool of previously-generated cuts is a finder
  - Only store globally-valid cuts
- MIP Solver must specify finder schedule, branching schedule

Calls to Cut Generation Library (Cgl)

- Takes an OsiSolverInterface as input
  - Gets all problem data from the interface
  - Assumes the point to cut off (x^*) is the current solution
    - Solver must look like it just solved the LP
- Fills a container with OsiRowCuts

PicoLPInterface

Inherits all of OsiSolverInterface

Added capabilities for branch-and-cut:

- Integrality management
  - OSI solver always thinks all variables are continuous (LP)
  - PICO overrides query methods (like isBinary(var)) for CGL
  - Issues with using OSI's setInteger()
    - LP-only solvers can consider that an error
    - Possible reset of internal data structures
- Other changes for row addition/deletion (Ojas will cover)
  - Track row numbers for loaded cuts
  - Explicit basis manipulation
- Currently support: CLP, Cplex, Soplex, glpk

PicoLPInterface Functionality - RestoreLP

- Make OsiSolverInterface solver look like it has just solved a problem
- Currently needed when
  - Start from an externally-computed (or saved) root solution
  - Cuts age out
  - Pseudocost (gradient) initialization
  - A cut finder modifies the OsiSolverInterface object

- Solver-independent, hopefully efficient method:
  - Reset the basis and bounds
  - Resolve

PicoRowCuts

- Sparse, pointer to an OsiRowCut
- One sided (upper bound)
- Sorted by column index (for dot product)
- Non-zero coefficients only on structural variables and core row slacks/artificials
- Use solver infinity consistently
- Hash value
- Reference count
- Age
- Persistence
- ID (short handle), cut finder, etc, for debugging

picoRowCuts - Hashing

- Store Hash value on construction:
- Compute Canonical Form for vector coefficients:
  - Round to given accuracy (default .01)
  - Scale so largest absolute value is 1
  - Must round first
- Hash value is hash of canonical form
- Parallel vectors should hash to the same value
- Store cut pool and loaded cuts in hash tables

Testing for Parallel Cuts

Test for redundancy or domination

- Hash values must be equal
- Must have same direction (sign on first element)
- Must have no angle between them (within tolerance):

$$\cos\ \theta_{12} = \frac{a_1 \bullet a_2}{\|a_1\|\ \|a_2\|} = 1$$

PICO Cut Finders

- Takes PicoLPInterface and a solution vector
- Returns an array of PicoRowCuts

- Explicitly signals infeasibility detection
  - Cgl uses infeasible cuts. Some solvers consider that an error.
- Wrapper class for Cgl generators
  - Interprets infeasibility
  - Eliminates redundancy for each call
  - Primitive global validity claims (safe defaults)
  - Corrections to avoid LP stomping
  - Substitutes for cut row slacks (pending)
  - Cut-finder-specific initialization (e.g. reducing output)

Pico Cut Finders

- A PICO cut finder knows classes of applicable MIPs
- At the start of the MIP computation, call cut finder with the problem representation
  - Cut finder determines whether it applies to this problem
    - e.g. checking for cutting/packing structure
  - Cut finder can set up data structures

Issue: Branch and Cut Context

- CGL cut finders have no notion of core rows vs. cuts (temporary)
- Generally view the current problem (bounds etc) as single problem

- Correctness issues
  - Nonzero coefficients on slacks of volatile rows (substitution)
- Possible efficiency issues
  - Global validity

Global Validity

Cuts like the TSP subtour elimination cuts are globally valid (apply to all subproblems).

- Can be shared

Recall a basic form for Gomory cuts (for binary problems):

$$x_i - \sum_{x_j \in x_L} \lfloor g_j \rfloor (x_j - \ell_j) - \sum_{x_j \in x_U} \lfloor g_j \rfloor (u_j - x_j) \geq \lfloor x_i^* \rfloor$$

X_L = variables at lower bound ℓ_j, X_U = variables at upper bound u_j

- CGL Gomory cut finder uses OsiSolver interface to get bounds
  - Resulting Gomory cuts only valid in subtree (bounds match)

Sandia National Laboratories

Global Validity

PICO maintains its own version of CGL Gomory cuts

- By using original binary status, makes globally valid cuts
  - Not necessarily a total improvement (denser)
  - Only globally valid if applied to globally valid rows

- We plan to add some parameters to tune
  - # cuts generated
  - Which rows considered

Currently no obvious way to pass parameters into CGL cut finders

- PICO can use start-up call

Cut Finder Quality Measure

- Each call to a cut finder has a quality measure based on cuts "credited" to the call (more later)
- Compute after LP resolve
- Quality of one cut = dual value times violation of old LP optimal
- Time = finder run time + (part of) LP solve
  - Forced nonzero
- One-run finder quality = $\frac{\sum \text{cut quality}}{\text{time}}$
- q_f quality of first run on a subproblem
  - Tracked for whole computation using exponential smoothing
- Quality for a single solve
  - Initialized to q_f + small factor if it loses to branching (grows with number of consecutive losses)

Eliminating cross-finder redundancy

Given sets of cuts from multiple finders (from same *x**), when we identify a pair of parallel cuts:

- Credit goes to the strongest cut, ties to the fastest (avg) finder
- Weaker cut eliminated unless its from the cut pool with positive reference count.
- If one finder proves infeasibility, it gets all the credit (+ bonus)

Sandia National Laboratories

Cut Finder Scheduling

- For the first few problems (default 10) sweep
  - All finders get a chance at the post-branch *x**
- After this first phase, zero-quality finders get small nonzero quality
- Competition phase
  - Proportional-share stride scheduling like PICO main scheduler

Incorporate cuts and resolve when:

- There are a lot of cuts waiting
- No finder is ready
- Ready finders are all much worse than ones run since last resolve

Proportional Share Scheduling: Simple Example

Job J_1, priority p_1 = 3. Job J_2, priority p_2=5

Interpretation: Dispatch job J_1 3 times for every 5 dispatches of J_2.

- Ticket u_i. Initialized to 0.
- Always run the job with the lowest ticket.
- After running increment ticket by $\frac{1}{p_i}$

J_1:	$\boxed{0}$	$\frac{1}{3}$	$\frac{1}{3}$	$\boxed{\frac{1}{3}}$	$\frac{2}{3}$	$\frac{2}{3}$	$\boxed{\frac{2}{3}}$	1	1
J_2:	0	$\boxed{0}$	$\boxed{\frac{1}{5}}$	$\frac{2}{5}$	$\boxed{\frac{2}{5}}$	$\boxed{\frac{3}{5}}$	$\frac{4}{5}$	$\boxed{\frac{4}{5}}$	1

Scheduling Cut Finders in Competition

Competition phase

- Proportional-share stride scheduling like PICO main scheduler
- Finders dispatched according to quality (*q*) and readiness (*r*)
- Dispatch finder with lowest ticket value
  - dynamic/delayed ticket computation

$$u_f + \frac{t}{r_f q_f}$$ where u_f is ticket before last run, t last runtime

When to Branch

Branching competes with the cut finders

- Quality
  - Let q be the expected bound movement (based on pseudocosts and solution value)
  - Let t be expected time for LP solve (save history with exponential smoothing)
  - Quality is $\dfrac{q+\varepsilon}{t}$
- Readiness is as a function that grows with *k*, the number of LP solves since the last branch (*b*-subscripted objects are weights)

$$\rho_b \max\{0, 1-\beta_b \exp(-\gamma_b k)\}$$

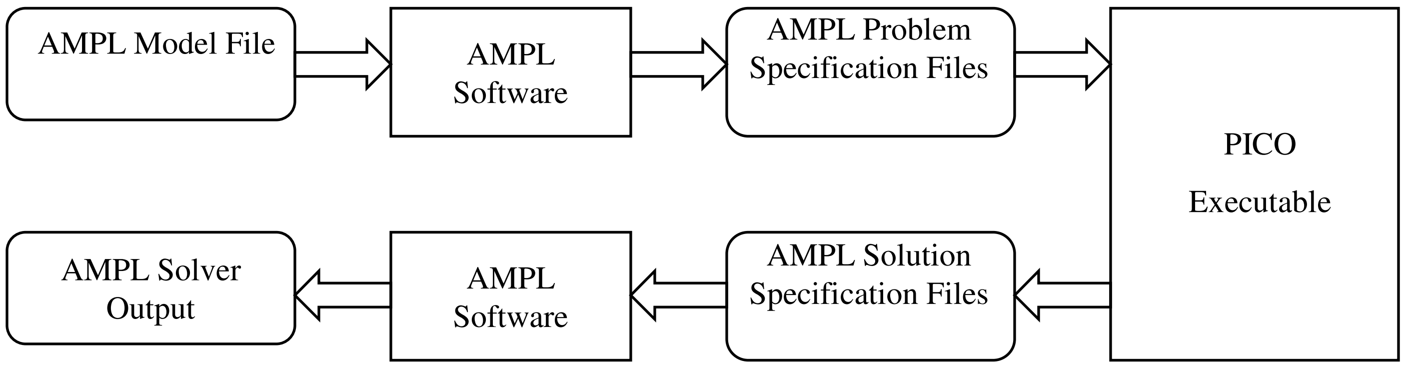
Sandia National Laboratories

Debugging Features

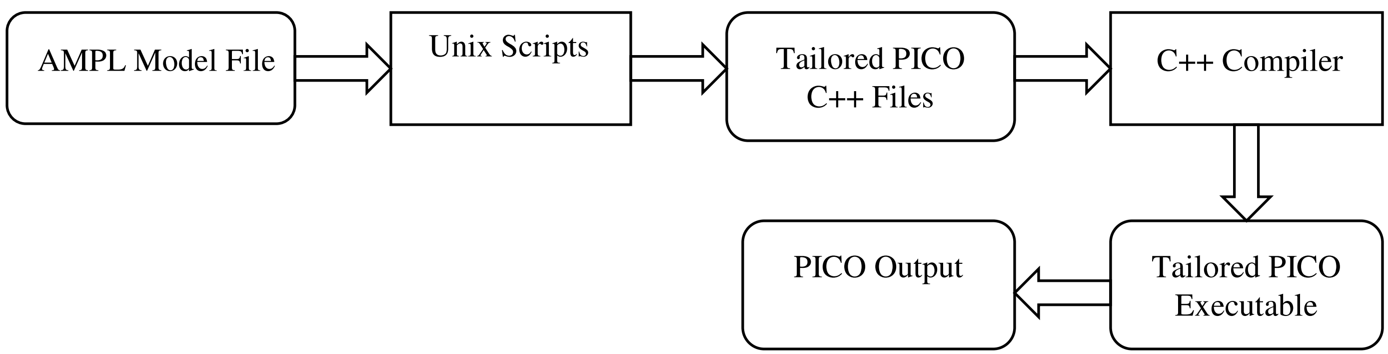
- Tracking watched points (e.g. known optima)
  - Throw an exception if
    - watched point violates an added global cut
    - Watched point violates an applicable local cut
- Pseudorandom timings

AMPL-PICO Interface

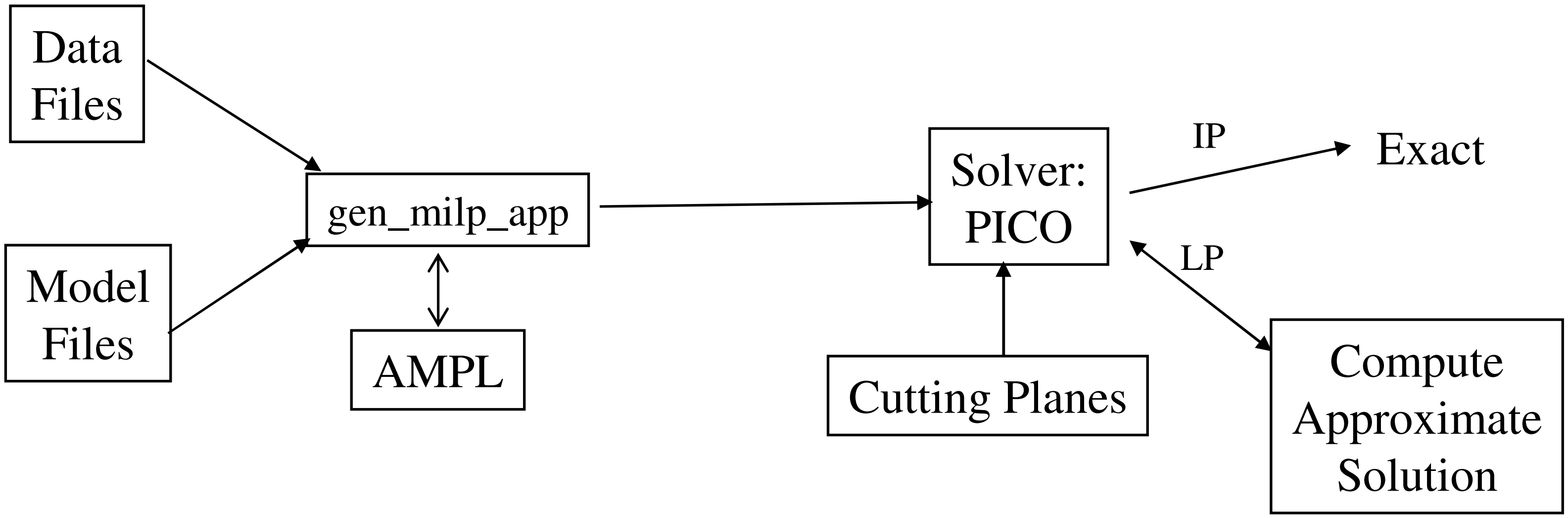
Standard AMPL interfaces



PICO AMPL Symbol Environment



AMPL-PICO Interface



- Write cutting-plane and approximate-solution code using AMPL variables
- Mapping transparent
- Announce cut finders to the driver object at start time.

Sandia National Laboratories

PICO CGL Cut Finder Defaults

- CGL cut finders currently enabled by default
  - Gomory cuts (PICO version)
  - MIR2
  - 2MIR
  - Flow cuts
  - Clique cuts
  - Probing (except on 64-bit architectures, etc)
- Others disabled for various reasons
  - Dominated
  - Needs special structure
    - pending verification or implementation of enforcement
  - Possibly producing errors/incorrect cuts
- Goal: command-line interface for enable/disable
- With user-defined cuts, almost all CGL cuts will be off by default

Next Steps for Cuts

- Throttling cuts
  - Adding all (except for redundancy filtering) is too slow
- Cut management parameter tuning
  - Try automated optimization
- Learn from MINTO/PARINO
  - Thanks Jeff Linderoth!