Coin LP
A tutorial

John Forrest

July 17 2006

Outline of Clp tutorial

Background

Some concepts

Example C++ code

Stand-alone solver

Less structured part:

- Q & A
- More examples
- Future -
  - What can I do for you?
  - What can you do for Clp

Background

Coin launched at ISMP 2000

Coin native Mps reader

Coin native factorization for Gomory cuts

Need for native code – first release of Clp 2002

- OSL on way out
- Clp - "reference code"? From OSL
- OSL influence but new mistakes

Slow improvements to code mainly for reliability

Use in Branch and Cut (see tomorrow's tutorial)

Some concepts

Target use is “meta” algorithms i.e. Repeated use of simplex.

Simplex oriented; weak in other areas - Q&A?

Virtual pivot choice - relatively easy for user to create own.

- Nice idea but steepest edge normally best
- Ideas to let user write simplex code – needs thought

Virtual matrix storage - easy for user to create own

- Can even do column generation or dynamic matrices
- Network matrix storage and factorization.
- Good example is Generalized Upper Bound coding

Many unfinished areas - “when I get time”

Classes

ClpModel - realization of OsiSolverInterface

- + names
- + virtual ClpMatrixBase
- Sub model constructor
- Const and non const array pointers

ClpSimplex – adds status arrays, factorization (could be virtual) and virtual pivot choice.

- ClpSimplexDual, ..Primal – no extra data, user does not need to know
- ClpSimplexNonlinear has SLP method and active set method

ClpInterior - ClpPredictorCorrector

More classes

ClpDualRowPivot – abstract class for choosing pivot row in dual

- ClpDualRowDantzig
- ClpDualRowSteepest – preferred

ClpPrimalColumnPivot – abstract class for in column in primal

- ClpPrimalColumnDantzig
- ClpPrimalColumnSteepest – preferred (and can be tuned)

ClpFactorization – uses CoinFactorization at present or ClpNetworkBasis if network

ClpNonLinearCost – piecewise linear objective – no phase

Matrix classes

ClpMatrixBase abstract class for storing matrix

- ClpPackedMatrix – pointer to CoinPackedMatrix plus bits
- ClpPlusMinusOneMatrix
- ClpNetworkMatrix – not integrated so slower than network code
- ClpGubMatrix etc – can be very fast but still not finished Q&A
- ClpSmallMatrix – will show to show amount of effort
  - Could be extended to be useful

Miscellaneous classes

Sophisticated users can derive from below for more control

- ClpEventHandler – iteration, factorization etc
- ClpMessageHandler – messages – also control printing

ClpObjective – abstract class for objective

- ClpLinearObjective – linear
- ClpQuadraticObjective – quadratic
- ClpUserCouldCreate – use with SLP or with more work with active set method

ClpSolve – to try and collect solution strategy in one place

Idiot – what can I say?

ClpPresolve – just an interface to CoinPresolve

Coin stuff

CoinFactorization – factorization code

- From some time ago
- Modified for extra sparsity coding
- Forrest-Tomlin update

CoinPresolve – used by ClpPresolve (and by OsiPresolve)

CoinMpsIO etc

CoinIndexedVector

CoinPackedMatrix

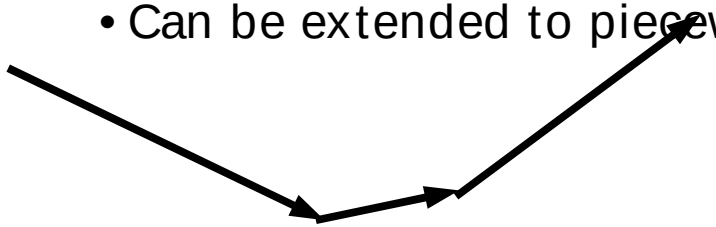
Simplex algorithms

Dual

- Very good description of what is in Clp dual -
  - Progress in the Dual Simplex Algorithm for Solving Large Scale LP Problems: Techniques for a fast and stable implementation. Achim Koberstein (koberstein@dsor.de)
- Artificial (increasing) bounds on variables to get dual feasible

Primal

- Still after all this time needs better pricing on long thin problems – but see example.
- Artificial (increasing) costs on infeasible variables – ClpNonLinearCost
  - Can be extended to piecewise linear objective



Documentation :-)

When I get time .....

As I said - very good description of what is in Clp dual -

- Progress in the Dual Simplex Algorithm for Solving Large Scale LP Problems: Techniques for a fast and stable implementation. Achim Koberstein (koberstein@dsor.de)

Use examples

- decompose.cpp
- dualCuts.cpp
- sprint.cpp
- This I can/will add to.

Ideas on how to improve things – wiki? - Q & A

First example - sprint

Example of way I think about building algorithms and using simplex

Originally developed for American Airlines crew scheduling problems

Same idea with variations used in most of my attempts to solve very large problems for IBM e.g. 15,285 rows 5,555,167 variables – 13 seconds.

- Get a feasible solution (possibly artificial)
- Fix part of problem so size and complexity much reduced
- Solve using simplex (normally primal)
- If good drop in objective value – repeat – else
- Go to normal simplex (so “algorithm” is finite)

Example assumes first few variables give feasible solution (no bounds)

- For real example see Clp/examples/sprint.cpp

So lets try something bigger

- This new data has 41,059,147 variables and 119,412 constraints!
  - Hours to download
  - Five minutes to read in
  - Four minutes to solve
- That was part of USA – so now we are going for whole of USA
  - An example of “medium”real world data exploding
    - Many people
    - Many skills
  - but then sensible algorithm brings it back down to plausible
- As an aside this is a case where problem gets more difficult as importance of individual decisions decreases – Tanker (ship) scheduling easy, planes harder, trucks harder, people harder .....
  - On the other hand exact optima less important

Generalized Upper Bounds

- Most (> 90%) of problem is non-overlapping constraints $\sum x_{ij} = b_i$
- If m rows then at most m basic columns so most GUB rows will just have one basic.
- So like simple upper bounds much of work is bookkeeping
- But needs factorization – but we can work with reduced basis
- Nice algorithm but delicate

$$\frac{A \quad B}{C \quad D} \qquad A - B \; D^{-1} \; C$$

Sprint approach

- Most (> 90%) of problem is non-overlapping GUB constraints
- If each such constraint has many members then candidate for sprint
- Change selection criterion to concentrate on subset of constraints
- If only one (basic) selected in a GUB constraint we can take out of small problem and recompute dual after solution
- So number of rows can be dramatically lower
- As with ordinary sprint surprising how few iterations

Ex OsiSimplexInterface
now all in OsiSolverInterface

- Misguided attempt to allow user to build an algorithm
- Now broken out so a solver says what it can do
    - None of the methods
    - Tableau stuff e.g. Updated row (Cplex, Clp)
    - Pivoting (Clp)
- I don't think best way to allow user to do it – but what is?
- Enough interest that I shouldn't just try and kill it.

Standalone Solver

- Fairly primitive – glad if someone would make more elegant
- Command line and/ or interactive
- Double parameters
- Int parameters
- Keyword parameters
- Actions
- Documented?
- Undocumented??
- Can produce reference list of parameters/ actions
  - Of course this uses an undocumented option :-)

Double parameters

- DualBound – initial fake “box” for variables
- DualTolerance – for reduced costs
  - Larger values can be faster in dual (Devex ratio effect)
- PreTolerance – infeasibilities in presolve less than this will be fixed up (rather than declared infeasible).
- PrimalTolerance – for primal infeasibilities
  - Larger values can be faster in primal
- PrimalWeight – initial extra cost for being infeasible
- Seconds – treat as maximum iterations after this time

Int parameters

- IdiotCrash – number of passes in idiotic crash
  - -1 primal makes up own mind, 0 off
- LogLevel – increases amount of printout (0==off)
- MaxFactor – maximum number of iterations between refactorizations – if default of 200 will compute
- MaxIterations – stop after this many iterations
- OutputFormat – for exporting model controls number of values per line and accuracy of values.
- Sprint - number of passes in sprint algorithm
  - -1 primal makes up own mind, 0 off

Keyword parameters (some)

- Direction – min, max, zero (also maximize as action)
- ErrorsAllowed – off,on – whether to allow errors in import
- KeepNames – on,off – whether to keep names after import
- Messages - off,on – whether to add Clpnnnn to messages
- Perturbation – on,off – whether to perturb problem
- Presolve – on,off – whether to do presolve
- PrintingOptions – normal, integer, all (+ others)
- Scaling – auto,off, equi, geo – whether to scale problem

Actions 1

- BasisIn file – reads in mps basis
- BasisOut file – creates mps basis
- Export file – creates mps matrix file
- Import file – reads in mps matrix file
- PrintMask mask – solution only prints names which match
- RestoreModel file – restores dumped model
- SaveModel file – dumps model to file
- SaveSolution file – saves solution in simple format
- Solution file (or stdout) – prints solution

Actions 2

- AllSlack – resets solution to all slack – for experimentation
- Barrier – not strong point – may bring up in Q&A
- DualSimplex
- Maximize
- Minimize
- PrimalSimplex
- Solve – for uniformity with Cbc
- UserClp – placeholder so user can modify Clpmain.cpp
- Stop, end, exit, quit

Undocumented stuff

- ObjectiveScale value – scale objective by this (in solve)
- RhsScale value – scale bounds etc by this (in solve)
- ReallyObjectiveScale value – scale objective in model
  - Can be -1.0 for stress testing or other for exporting
- ReallyScale – scale model (not just in solve)
- PassPresolve, preOpt, substitution – presolve tuning
- Dualize – make and solve dual model (experimental)
- PertValue – fine tuning of perturbation
- SpecialOptions – as in ClpSimplex.hpp
- Network, plusMinus – massage matrix for speed

Five minute contest

- Suggested model – Data/miplib3/dano3mip
  - But you can choose another one
- Run with just “clp file” gives presolve and dual
- Try options – e.g.
  - -presolve off
  - -dualTolerance 1.0e-6
  - -crash
- Fastest buys me a drink
- Partly for a break

Code generation ?

- Standalone solver makes it easy to experiment and find fast way of solving problem
- But what if you want to build model rather than read an mps file?
- Or what if you want to set a parameter you can find in ClpSimplex.hpp but not in solver?
- Up to now it was difficult to transfer settings but ...
- Cpp option – use it before the primal/Simplex or dual/Simplex and a file user_driver.cpp will be produced.
- The Makefile in Clp/examples can be used.

Interior Point

- Not my strong point – about as good as OSL's
  - Needs a bit more work on crossover to simplex
- Solves QPs as well
  - Solves most of test sets but can use too much memory
  - Crossover not implemented yet
- No reasonable native Cholesky ordering
  - Use Anshul Gupta's WSSMP package or
  - Use AMD or CHOLMOD code from U Florida
- So-so Cholesky factorization so use above
- Help!

Quadratic objectives

- Even less of a strong point – about as good as OSL's
  - Active set method
  - Not really quadratic – any nonlinear objective if methods coded
  - May do big push – but is it needed?
- Sequential Linear Program method also in
  - Robust – often best method is to do some passes with SLP and then go to Quadratic Simplex (often 0 iterations)
  - Again not restricted to quadratic
- Help! Q & A point – how comprehensive should CLP be?

CoinModel

- Designed to be flexible and fastish way of building a model
- addRow and addColumn
- setElement(i,j,value)
- getElement(i,j) (also by name)
- setRowLower etc etc
- Iterate over row or column
- Symbolic values
- Example - Clp/examples/addRows.cpp

More examples

- Internals of Clp – how to create your own matrix class
  - Reduce storage and increase speed?
  - ClpPackedMatrix has more than needed for many cases so creation is simpler than you think
- How to decompose a matrix and do Dantzig Wolfe
  - Example of using Clp (applicable to Osi)
- Whatever you suggest and I will try and describe how I would go about it ?

Matrix class needs

- Constructors etc
  - Default, from CoinPackedMatrix and other useful
  - =, clone and destructor
- Times, transposeTimes, sophisticated transposeTimes and subsetTransposeTimes
- CountBasis and fillBasis for creating basis
- Unpack, unpackPacked - unpack one column
- Add – add a column into a CoinIndexedVector
- RangeOfElements – largest smallest in matrix
- Extra needed if scaling will be used

Future of Clp

What is missing and should it be in?

- For experimentation
- For heavy duty use
- For teaching

What should be improved – prioritize?

- As above

Redesign to make it easier to replace?

Replacement factorization project?

Matrices for speed

Simplex codes I have known

Name	Year?	Contrib	Comments
None	1966	100	Paper tape generalized gub
LP 90/94	1967	0	First real code – tape – used QP
Alligator	1967	1	Over designed – tape
Umpire	1969	25	Very influential – drum (so F-T)
Sciconic	1974	High	Clean rewrite of Umpire – mainly in memory
Lamps	1980	100	First code for mini-computers?
None	1982	100	LP code for BBC micro!
Lantern	1983	100	LP code for microcomputers
SQL-LP	1986	100	Failure to do modeling/solving
MPSX/370	1987	Small	Vector processing
YKTLP	1988	100	First code > 32000 rows
OSL	1989	50	Commercial extension of YKTLP
Child-of-OSL	1997	100	Attempt at parallel code
Clp	2002	High	
Child-ofClp	2007	Small	Greatest Code ever written

Referenced code

Sprint – Clp/examples/sprint.cpp

- sprintEasy2.cpp and sprintEasy.cpp (with nw04a)

ClpSmallMatrix.?pp and testSmall.cpp in Clp/examples

addRows.cpp in Clp/examples

For some help – clp -verbose 11 -?