Experience with a Parallel Branch, Cut and Price framework
Features

- Generalized Branching objects:
  - Change bounds of variables/constraints
  - even \( y_i = 0 \), \( \sum a_i x_i \leq b \) or \( y_i = 1 \)

- Use of Cut and Variable pools:
  - Locally valid cuts (Globally valid as special case)
  - Variable pool

- User "Hooks" (mostly C++ virtual objects)
  - LP Solver Class
    - Solve LPs
    - Add/Delete Rows/Columns
    - Modify bounds
    - Return/accept warmstart information
  - Message Passing Protocol
  - Cut/Variable generation
  - Branching Object generation
  - Logical fixing
  - ........
Flow at LP node

- Wait for LP formulation
- Solve LP

LP feas < UB
- Generate Columns and/or Cuts
- Branch ?
  - yes: Send Branching info to TM
  - no: Dive?
- yes: New Cols ?
  - yes: Send "fathom" to TM
  - no: no

LP infeas > UB
- Generate Columns
- Propagate new UB
- yes: no
Overview

- LP formulation based:
  - Lower bounding by solving LPs
  - Any LP solver (not necessarily Simplex based)

- Parallel:
  - Master / Slaves model.
  - Distributed Network.

- Branch

- and Cut (constraint generation):
  - Strengthen LP formulation
  - Dynamically use constraints in subproblems.

- and Price (variable generation):
  - Include new variables in LP formulation
  - Dynamically use variables in subproblems.

- Framework:
  - User has to provide/select problem specific parts.
swath

- **Initial statistics:**
  - 885 rows, 6805 variables of which 6724 0-1, 1 for GAMS objective and 80 continuous.
  - Continuous solution 334.4968581, best known integer solution 497.603.

- **Initial analysis:**
  - 80 continuous come in 20 groups of 4 identical variables - so we can reduce to 20 variables - $x_i$
  - We can introduce 380 $y$ accumulation variables so $y_{ij} = \sum z_{ijk}$ (but $\sum z_{ijk} \leq 1$ so $y_{ij}$ a 0-1 variable).
  - $x_i$ have no cost
n=20

20 \ y_{ij} + x_i - x_j \leq 19 \ for \ i=0...19, \ j=0...19 \ i!=j

where \ y_{ij} \ are \ 0-1 \ and \ x_i \ are \ "continuous"

these are only constraints where x appears

not only are y 0-1 but \ \Sigma y_{ij} \leq 1 \ for \ all \ i

If \ x_j \geq x_i+1 \ then \ y_{ij} \ can \ be \ 1 \ (and \ y_{ji} \ is \ 0)

If smallest x is >0 then all x can be adjusted.

Given order of x, values of x can be adjusted so that x takes on values 0...19 exactly once.

So x variables are really general integer and getting them correct forces correct solution!
20 \( y_{ij} + x_i - x_j \leq 19 \)
20 \( y_{ji} + x_j - x_i \leq 19 \)

\( y_{ij} + y_{ji} \leq 1 \)

for any \( i,j,k \)
\( y_{ij} + y_{ji} \leq 1, y_{ik} + y_{ki} \leq 1, y_{jk} + y_{kj} \leq 1 \)
\( y_{ij} + y_{ik} \leq 1, y_{ji} + y_{jk} \leq 1, y_{ki} + y_{kj} \leq 1 \)

Only way three \( y \) can be one is \( y_{ij} = y_{jk} = y_{ki} = 1 \) or similar

\( x_j \geq x_i + 1, \ x_k \geq x_j + 1, \ x_i \geq x_k + 1 \rightarrow \)
\( x_i \geq x_i + 3 \) !! - so

\( y_{ij} + y_{ji} + y_{jk} + y_{kj} + y_{ki} + y_{ik} \leq 2 \)
swath

- Original:
  - Continuous solution 334.4968581, best known solution 497.603.

- Previous cuts can be generalized

- With cuts and reformulation and $y_{ij}$ in SOS
  - Continuous 461.8312
  - Proven best solution 467.4075
  - 448 nodes without any further cuts

- Key ideas
  - Simplify so can see structure
  - Deduce $x$ relationships
  - Use for powerful cuts
Set covering problem

Initial statistics:
- 4,944 rows and 1,372 columns.
- All 0-1, objective is all 1.0
- Continuous solution 403.8465, best known solution 423.

Initial analysis:
- Can be reduced to 4,323 rows and 882 columns.
Cuts
- Odd hole cuts
- Prime cover cuts (Bellmore-Ratcliff)
- ..... 
- [Disjunctive cuts] - should have tried

Branching
- Close to half (and long columns)
  - $x_i = x_j = 1$ or $x_i + x_j \leq 1$
  - Slack branching cut
  - On tight constraints - $x_1 = 1$ or $x_1 = 0 \& x_2 = 1$ or ..
  - ..... 

Parallel on 30 machines - was obvious would take too long.
- We intend to keep trying
Initial statistics:
- 3411 rows, 5325 variables of which 5323 0-1.
- Continuous solution -611.85, best known solution -553.75.

IBM solution for steel mill planning

Initial analysis (RC 21071):
- $\sum_{i \in N_j} O^i x^i_j \leq W_j z_j \quad 1 \leq j \leq M$
- $\sum_{j \in N_i} x^i_j \leq 1 \quad 1 \leq i \leq N$
- $\sum_{c \in C_j} y^c_j \leq 2 \quad 1 \leq j \leq M$
- $x^i_j \leq y^{c(i)}_j \quad 1 \leq i \leq N, \quad 1 \leq j \leq M$

M = 24 (slabs) and N = 439 (orders)

Knapsacks do not overlap
Column Generation approach:

- Master has 2 continuous variables (purely for reporting) and 24 z variables
- 24 ex-knapsack constraints $\sum_{q \in Q} q_{ij} \leq z_i$
- 439 constraints $\sum_{j \in N} \delta_{qij} q_{qij} \leq 1$ where $\delta_{qij}$ is 1 if $x_{ij}$ is included in $q_{qij}$

Each knapsack

- $\sum_{i \in N_j} O^i x^i_j \leq W_j$
- $\sum_{c \in C_j} y^c_j \leq 2$
- $x^i_j \leq y^{c(i)}_j$ for $1 \leq i \leq N$
Brute force is possible:

- Number of x variables in a knapsack varies from 7 to 290.
- Total number of q variables is only 604,333

Results:

- Total generation and solution time 4 seconds.
  Integer LB is -563.846 as against original of -611.85.
- Continuous solution is integer so proven optimal solution of -563.846

Bad/good news:

- IBM problem - customer says - well that was only a toy problem - can you solve this?
Statistics:
- 68,122 rows, 62,394 variables of which 62,392 0-1.
- Continuous solution -1200.947
- Column generation master 9,560 rows and initial 0-1 solution of 0.0

M = 74 and N = 9,483
Not quite as easy!
- Knapsacks still do not overlap
- Some y variables switch on 700 x variables and 700!10 proposals in some knapsacks
Approach:

- Most large knapsacks
  - Use simple greedy heuristic
  - Close to LP relaxation
- Small knapsacks
  - Enumerate once and keep in pool
- Remaining knapsacks
  - Use simple heuristic - but
  - Not close to LP relaxation - needs better heuristics
- Looks as if gives close to optimal answers - more work needed on getting LB.
**mkc7 - results**

Effect of full enumeration of knapsacks on solution quality

<table>
<thead>
<tr>
<th>enumerate if size of knapsack</th>
<th>variables generated</th>
<th>Best possible LB</th>
<th>Best LP solution found</th>
<th>Best IP solution found</th>
<th>Time to best IP (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1</td>
<td>0</td>
<td>-1186.115</td>
<td>-1177.489</td>
<td>-1160.723</td>
<td>34.31</td>
</tr>
<tr>
<td>&lt;30</td>
<td>1678</td>
<td>-1186.038</td>
<td>-1177.685</td>
<td>-1171.171</td>
<td>77.48</td>
</tr>
<tr>
<td>&lt;50</td>
<td>104723</td>
<td>-1185.968</td>
<td>-1182.967</td>
<td>-1178.163</td>
<td>70.81</td>
</tr>
</tbody>
</table>

Original formulation had LB of -1200.947
(and took 101 seconds to continuous optimum!)
Two problems solved - third by end year?

Column generation
- Not totally symmetric with cut generation (especially when exact continuous optimum can not be found)
- Theoretically can do cuts and price on same problem
- Good to get solutions early
- Need column generation in parallel in early phases

Flexible branching objects
- Mixed variable/constraint branching
- Easier valid branching with column generation

Thinking beats computing power

Exploring ways to make framework more available