Column Generation in GAMS

Extending the GAMS Branch-and-Cut-and-Heuristic (BCH) Facility

Michael R. Bussieck

GAMS Software GmbH
GAMS Development Corp

MBussieck@gams.com

83rd Working Group Meeting
Real World Optimization

Mathematical Optimization in Transportation - Airline, Public Transport, Railway -
<table>
<thead>
<tr>
<th>Agenda Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>枝分割と動的計画問題</td>
</tr>
<tr>
<td>最初の例</td>
</tr>
<tr>
<td>BCHを用いた制約条件の生成</td>
</tr>
<tr>
<td>BCHを用いた土丁目の生成</td>
</tr>
</tbody>
</table>
Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Constraint Generation with BCH
- Column Generation with BCH
Modeling Systems

- Best way to model and solve optimization problems
- Solid foundation based on “Separation”
  - Separation of Model and Data
  - Separation of Model and Algorithm
- Art of Modeling
- Some Modeling Systems provide (all) features of a programming language (e.g. GAMS, MOSEL, …)
  - Avoid usual stumbling blocks of programming
  - Integration of optimization models
- Solver is black box
- Good approach for >95% of optimization problems
- Small number of models/users that need/want more
  - Solver/User information exchange to guide/improve the solution process.
Solution Frameworks

- Branch-and-Cut(-and-Price)
  - Abacus, MINTO
  - BCP, Bonmin, Cbc, SCIP, Symphony, …
  - Cplex, Xpress-MP, …

- Required Knowledge for Implementation
  - IT knowledge (C/C++/JAVA, Solver APIs)
  - Mathematical programming knowledge
  - Application specific knowledge

- Utilize rapid prototyping capability for improving solution process by user supplied information (cuts, heuristics, …)
“Classical” Branch-and-Cut-and-Heuristic

• Cut Generator and Heuristic
  – Represented in terms of original GAMS problem formulation
  – Independent of the specific solver
  – Use any other model type and solver available in GAMS in
Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Constraint Generation with BCH
- Column Generation with BCH
Multi-Knapsack

http://www.gams.com/modlib/libhtml/bchmknap.htm

Binary variables \( x(j) \); Positive variables \( \text{slack}(i) \);
Equations \( \text{mk}(i), \text{defobj} \); Variable \( z \);

\[
\text{defobj}.. \quad z = e = \sum(j, \text{value}(j) \times x(j));
\]

\[
\text{mk}(i) .. \quad \sum(j, a(i,j) \times x(j)) = \text{l} = \text{size}(i);
\]

model m /all/; solve m max z using mip;

Separation Problem for Cover Cuts:
\[ z.l < 1 \]

Cover Cuts \( c(j)=y.l(j) \):
\[ \sum(c(j),x(j)) = \text{l} = \text{card}(j)-1; \]

Binary variable \( y(j) \) membership in the cover;
Equations \( \text{defcover}, \text{defobj} \); Variable \( z \);

\[
\text{defobj}.. \quad z = e = \sum(j, (1-x.l(j)) \times y(j));
\]

\[
\text{defcover} .. \quad \sum(j, a(i,j) \times y(j)) = \text{g} = \text{size}_i+1;
\]

model cover /all/; solve cover min z using mip;
Cover Cuts and Rounding Heuristic

- Activate BCH facility (option file):
  
  ```
  usercutcall  mknap
  userheurcall  mknap –heuristic=1
  ```

- Separation model:
  
  ```
  Excute_loadpoint ‘bchout’; // Get node solution from solver

  * Cover cut:
  If (z.l<1, numcuts = 1;
     x_c(‘1’,j) = y.l(j);
     rhs_c(‘1’) = sum(j, y.l(j)) - 1;
     sense_c(‘1’) = 1); // 1 =l=, 2 =e=, 3 =g=

  * Heuristic
  rhs(i) = rhs(i) - sum(j$(x.l(j)=1), a(i,j));
  loop(j$(x.l(j)<1),
     if (smin(i, rhs(i)-a(i,j))>=0,  x.l(j) = 1; rhs(i) = rhs(i) - a(i,j);
     else x.l(j) = 0));
  ```
## Cplex Log with BCH Active

<table>
<thead>
<tr>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Best Node</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4134.0741</td>
<td>2</td>
<td></td>
<td>4134.0741</td>
</tr>
<tr>
<td>***** Calling heuristic. Solution obj:** 3300.0000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 0+ 0</td>
<td>3300.0000</td>
<td>4134.0741</td>
<td>3</td>
<td>25.27%</td>
<td></td>
</tr>
<tr>
<td>***** Calling cut generator. Added 2 cuts**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 3871.4286 2</td>
<td>3300.0000</td>
<td>User: 1</td>
<td>5</td>
<td>17.32%</td>
<td></td>
</tr>
<tr>
<td>***** Calling heuristic. obj = 3300**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>***** Calling cut generator. Added 1 cut**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 0 3800.0000 3</td>
<td>3300.0000</td>
<td>User: 1</td>
<td>7</td>
<td>15.15%</td>
<td></td>
</tr>
<tr>
<td>***** Calling heuristic. obj = 3300**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>***** Calling cut generator. No cuts found**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>***** Calling heuristic. obj = 3300**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>***** Calling cut generator. No cuts found**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>***** Calling heuristic. obj = 3300**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 2 3800.0000 3</td>
<td>3300.0000</td>
<td>3800.0000</td>
<td>8</td>
<td>15.15%</td>
<td></td>
</tr>
<tr>
<td>***** Calling cut generator. No cuts found**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>***** Calling heuristic. obj = 3800**</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* 1 0 integral 0</td>
<td>3800.0000</td>
<td>3800.0000</td>
<td>9</td>
<td>0.00%</td>
<td></td>
</tr>
</tbody>
</table>
Oil Pipeline Design Problem

• Real Example: Oil Pipeline Design Problem
  – Cuts generated when new incumbent is found
  – Rounding Heuristics, Local Branching

• Performance Improvements
  – Cplex/BCH: 20 minutes
  – Regular Cplex: 450 minutes

• Overhead of BCH
  – Time spent within the callback functions minus MIP computation on cuts and heuristics: 20% ~ 25%
Some Recent/Ongoing Extensions

• Features
  – *Cuts and Heuristics*
  – Incumbent Filters
  – Pricing/Branching (Haase, Lübbecke, Vigerske)
  – Thread safe, BCH in a library

• Scope of Application
  – *Implement user heuristics/cuts for special problems*
  – Rapid Prototype Development for Algorithmic Ideas
    • LPEC (Michael Ferris, U Wisconsin)
    • RINS for MINLPs (Stefan Vigerske, HU Berlin)
    • Quesada/Grossmann Algorithm for MINLP
  – Constraints/Column Generation on the fly
Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Constraint Generation with BCH
- Column Generation with BCH
Traveling Salesman Problem

• Start with “matching constraints”:
  \[ \sum_{i} x(i,j) = 1 \] for all \( j \)
  \[ \sum_{j} x(i,j) = 1 \] for all \( i \)

• Add subtour elimination constraints when they are needed:
  \[ \sum_{i,j \in S} x(i,j) < \text{card}(S) - 1 \]

Repeat
  
  solve tsp min obj using mip;
  if (subtours, add cut);
  until no subtours  
  (GAMS Model Library tsp1-tsp5)
BCH Implementation of TSP

- Perform regular B&C
  - start with matching constraints
  - presolve has to be turned off!

- Incumbent Accept/Reject Facility (userincbcall)
  - check for subtours
  - if rejected, store subtour elimination constraint

- Cut Facility (usercutcall)
  - supply subtour elimination constraint
Cplex Log for TSP

Root relaxation solution time = 0.00 sec.
*** Calling cut generator. /
*** Checking incumbent with objective 20. Rejected!

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Cuts/</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Node</strong></td>
<td><strong>Left</strong></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*** Calling cut generator. Added 3 cuts
*** Checking incumbent with objective 136. Rejected!
*** Calling cut generator. /
*** Checking incumbent with objective 78. Accepted!

* 0 0 integral 0 78.0000 Cuts: 13 4 0.00%
0 0 cutoff 78.0000 78.0000 4 0.00%
Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Constraint Generation with BCH
- Column Generation with BCH
Graph Coloring Algorithm

• Given graph \((V,E)\), find coloring \(c\) of \(V\) with \(c(u) \neq c(v)\) for all edges \(e = uv\) in \(E\).

• Mehrotra/Trick:
  \[
  \min \text{sum}(i, x(i)), \text{ st. sum}(i \text{ contains } v, x(i)) \geq 1, \text{ i independent set of } (V,E)
  \]

Pricing problem:
Find an independent set \(j\) with \(\sum(v \text{ in } j, \pi_v) > 1: \max \sum(v, \pi(v)z(v)), \text{ st. } z(u)+z(v) \leq 1, uv \text{ in } E, z \text{ binary}\)

Repeat
  solve master min obj using rmip;
  solve pricing max obj2 using mip;
  if (obj2.l>1, add column);
  until obj2.l<1;
  solve master min obj using mip;
BCH Implementation of Graph Coloring

- Disadvantage of traditional GAMS implementation
  - Regeneration of master problem
  - Warm versus hot start of master LP
- BCH Solution:
  - Start with a trivial independent set covering
  - Solve restricted master and inside solver call pricing facility *(userpricingcall)*

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Objective</th>
<th>In Variable</th>
<th>Out Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.252941</td>
<td>x91</td>
<td>x83</td>
</tr>
<tr>
<td>2</td>
<td>3.252941</td>
<td>defcover(20)</td>
<td>slack</td>
</tr>
</tbody>
</table>

LP status(1): optimal
Optimal solution found.
Objective : 3.252941
--- Calling pricing problem.
3 columns added.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Objective</th>
<th>In Variable</th>
<th>Out Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.252941</td>
<td>x92</td>
<td>defcover(20)</td>
</tr>
<tr>
<td>2</td>
<td>3.250000</td>
<td>x80</td>
<td>slack</td>
</tr>
</tbody>
</table>

LP status(1): optimal
Optimal solution found.
Objective : 3.250000
Outlook: Branch-and-Price

Repeat
   solve master min obj using rmip;
   solve pricing max obj2 using mip;
   if (obj2.l>1, add column);
until obj2.l<1;
solve master min obj using mip;

• True Branch-and-Price:
  – Column generation at the nodes of the B&B tree
  – Branching rule that is compatible with pricing problem
  – Prototype written in GAMS (tree manager, with Haase, Lübbecke)
  – Use solver framework (e.g. SCIP) to implement Branch-and-Price with BCH

does not guarantee optimal solution
Summary

- Solver independent BCH readily available with GAMS

- Implement user heuristics, cuts and dynamic constraints/columns without too much computer science knowledge in your problem namespace

- Build rapidly prototypes of advanced algorithms in little time concentrating on the essential ideas

- Ongoing project: BCH for Branch-and-Price

http://www.gams.com/docs/bch.htm (documentation)
http://www.gams.com/modlib/libhtml/alfindx.htm (examples)