GAMS
Branch-and-Cut & Heuristic Facility

Michael R. Bussieck
MBussieck@gams.com

GAMS Software GmbH
GAMS Development Corp

INFORMS Annual Meeting
San Diego 14 October 2009
Welcome/Agenda

Branch-and-Cut & Heuristic Facility
First Example
Extensions and Open Source
Implicit Constraints with BCH
<table>
<thead>
<tr>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Branch-and-Cut &amp; Heuristic Facility</strong></td>
</tr>
<tr>
<td><strong>First Example</strong></td>
</tr>
<tr>
<td><strong>Extensions and Open Source</strong></td>
</tr>
<tr>
<td><strong>Implicit Constraints with BCH</strong></td>
</tr>
</tbody>
</table>
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
- Extensions and Open Source
- Implicit Constraints 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 \( z \);

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

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

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

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

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

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

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

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

```
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:

```plaintext
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);  // cut matrix

* 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>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>*** Calling heuristic. Solution obj: 3300.0000</td>
</tr>
<tr>
<td>* 0+ 0</td>
</tr>
<tr>
<td>*** Calling cut generator. Added 2 cuts</td>
</tr>
<tr>
<td>0 0 3871.4286 2</td>
</tr>
<tr>
<td>*** Calling heuristic. obj = 3300</td>
</tr>
<tr>
<td>*** Calling cut generator. Added 1 cut</td>
</tr>
<tr>
<td>0 0 3800.0000 3</td>
</tr>
<tr>
<td>*** Calling heuristic. obj = 3300</td>
</tr>
<tr>
<td>*** Calling cut generator. No cuts found</td>
</tr>
<tr>
<td>*** Calling cut generator. No cuts found</td>
</tr>
<tr>
<td>*** Calling heuristic. obj = 3300</td>
</tr>
<tr>
<td>0 2 3800.0000 3</td>
</tr>
<tr>
<td>*** Calling cut generator. No cuts found</td>
</tr>
<tr>
<td>*** Calling heuristic. obj = 3800</td>
</tr>
<tr>
<td>* 1 0 integral 0</td>
</tr>
</tbody>
</table>
Oil Pipeline Design Problem

• Real Example: Oil Pipeline Design Problem
  – Cuts generated when new incumbent is found
  – Rounding Heuristic, 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%
Oil-Design (Convergence)
Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Extensions and Open Source
- Implicit Constraints with BCH
Some Recent/Ongoing Extensions

• Features
  – *Cuts and Heuristics*
  – *Incumbent Filters*
  – Branching (Alexander Martin, TU Darmstadt)
  – Column Generation (Knut Haase, TU Dresden)
  – 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
  – Implicit constraints
BCH and Open Source

- Open Source Solvers aware of BCH
  - COIN-OR’s Cbc, COIN-OR’s Bonmin, ZIB’s SCIP

- Open Source codes
  - Highly flexible. E.g. callbacks for solving node problem
  - E.g. SCIP: General constraint handler

- Commercial MIP codes
  - Build for maximum performance without user interaction
  - Restricted user interaction. For example, Cplex:
    - No cut callbacks at infeasible nodes
    - Disabled dynamic search when user callbacks active
<table>
<thead>
<tr>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch-and-Cut &amp; Heuristic Facility</td>
</tr>
<tr>
<td>First Example</td>
</tr>
<tr>
<td>Extensions and Open Source</td>
</tr>
<tr>
<td>Implicit Constraints with BCH</td>
</tr>
</tbody>
</table>
Traveling Salesman Problem

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

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

Repeat

solve tsp min obj using mip;
if (subtours, add cut);
until no subtours  \hspace{1cm} (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>Node</td>
<td>Left</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. Added 2 cuts
*** 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%
Summary

• BCH readily available with GAMS

• Implement user heuristics, cuts and implied constraints without too much computer science knowledge in your problem namespace

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

• Use unified interface to interact with different B&C frameworks

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