Global Optimization and the GAMS Branch-and Cut Facility

Hernan Abeledo, Michael Bussieck, Leon Lasdon, Alex Meeraus, Hua Ni

George Washington University
GAMS Development Corporation
University of Texas
GAMS Development Corporation
George Washington University

CORS/INFORMS Joint International Meeting Banff
May 16-19 2004
Branch-and-Cut (B&C)

- MIP problems benefit from *user supplied* cutting planes and integer solutions in a B&C algorithm
- Implementation facilities:
  - MIP solver callback functions (CPLEX, XPRESS, ...)
  - B&C framework (ABACUS, COIN BCP, ...)
- Required Knowledge for Implementation
  - IT knowledge (C/C++/JAVA, Solver APIs)
  - Mathematical programming knowledge
  - Application specific knowledge
- Supply GAMS users with an easy access to B&C: 
  GAMS Branch-and-Cut & Heuristic (BCH) Facility
BCH Unique Facility

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

- Design Principle:

![Diagram of BCH Unique Facility]
Example: Multi-Knapsack

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

Binary variables $x(j)$; Positive variables $slack(i)$;
Equations $mk(i)$, $defobj$; Variable $z$;

$$defobj.. \quad z =e= \sum(j, value(j) \cdot x(j))$$
$$mk(i).. \quad \sum(j, a(i,j) \cdot x(j)) =l= size(i);$$

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

The original model formulation

Separation Problem for Cover Cuts:
$z.l<1$

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

Binary variable $y(j)$ membership in the cover;
Equations $defcover$, $defobj$; Variable $z$;

$$defobj.. \quad z =e= \sum(j, (1-x.l(j)) \cdot y(j))$$
$$defcover.. \quad \sum(j, ai(j) \cdot y(j)) =g= size_{i+1};$$

model cover /all/; solve cover min $z$ using mip;
Cover Cut Separation Model

- Master model (bchmknaps.gms):
  ```
  execute_unload 'multiknapdata', j, l, a, size
  // Export of data
  ```

- Activate Cplex/BCH facility (cplex.opt):
  ```
  usercutcall bchcover.gms
  ```

- Separation model (bchcover.gms):
  ```
  $gdxin multiknapdata
  // Get data from original model
  $load j i a size
  $gdxin bchout
  // Get fractional solution from solver
  $load x
  $include cutmodel
  // model definition and solution
  * Define additional cut for original problem:
  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=
  ```
**CPLEX with BCH Log**

<table>
<thead>
<tr>
<th>Node</th>
<th>Left</th>
<th>Objective</th>
<th>IInf</th>
<th>Best Integer</th>
<th>Cuts/</th>
<th>Best Node</th>
<th>ItCnt</th>
<th>Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>4134.0741</td>
<td>2</td>
<td>4134.0741</td>
<td>3</td>
<td>4134.0741</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- ***** Calling cut generator. Added** 1 cut
- **4090.6542** 2
- **User:** 1 4
- ***** Calling cut generator. Added** 1 cut
- **4001.4270** 4
- **User:** 1 5
- ***** Calling cut generator. Added** 1 cut
- **3950.4202** 4
- **User:** 1 6
- ***** Calling cut generator. Added** 1 cut
- **3871.4286** 2
- **User:** 1 7
- ***** Calling cut generator. Added** 1 cut
- **3841.6244** 6
- **User:** 1 9

- ***** Calling cut generator. No cuts found**
- ***** Calling cut generator. No cuts found**
- **1 1 3800.0000** 3
- **3800.0000** 10

- ***** Calling cut generator. Added** 1 cut
- ***** Calling cut generator. No cuts found**
- **User:** 1

- ***** Calling cut generator. No cuts found**
- **2 1 0** 3800.0000 3800.0000 12 0.00%

- ***** Calling cut generator. No cuts found**
• Real Example: Oil Pipeline Design Problem

• 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%
Convergence – Pipeline Design
OK, but what has all this to do with Global Optimization?

- Use any other model type and solver available in GAMS
- Solve difficult MIP problems using GO for cut separation
Difficult Network Problem

- Single-commodity, uncapacitated, fixed-charge network flow problem

\[
\min \sum_{(i,j) \in A} (f_{ij}y_{ij} + c_{ij}x_{ij}), \text{s.t. } \sum_{(j,i) \in \delta^-(i)} x_{ji} - \sum_{(i,j) \in \delta^+(i)} x_{ij} = b_i, 0 \leq x_{ij} \leq M y_{ij}, y \in \{0,1\}
\]

- Problem class includes Steiner Tree Problem
- 83 instances (including 24 not proven optimal)
Dicut Inequalities

- Dicut: \[ \sum_{(i,j) \in \delta^-(S)} y_{ij} \geq 1 \] if \( S \subseteq V \) and \( b(S) > 0 \).

- Separation problem:
  \[ \xi = \min \left\{ \sum_{(i,j) \in A} \bar{y}_{ij}z_j(1 - z_i) : \sum_{i \in V} b_i z_i > 0, \ z_i \in \{0, 1\} \ \forall \ i \in V \right\} \]

- Non-convex quadratic binary program:
  - Ortega/Wolsey: Greedy algorithm
  - Let’s solve this exactly (small, \#variables |V|)
Global versus Local

- Berlin52 – from SteinLib
  - 52 nodes (1 source, n’=15 sinks), 1326 edges

<table>
<thead>
<tr>
<th></th>
<th>CPLEX (no BCH)</th>
<th>BCH + SBB (local)</th>
<th>BCH + BARON (global)</th>
</tr>
</thead>
<tbody>
<tr>
<td># cuts</td>
<td>0</td>
<td>139</td>
<td>610</td>
</tr>
<tr>
<td># nodes</td>
<td>168449</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Time</td>
<td>1000</td>
<td>1000</td>
<td>223</td>
</tr>
<tr>
<td>Gap</td>
<td>45%</td>
<td>39%</td>
<td>0%</td>
</tr>
</tbody>
</table>
Heuristics

- Local Branching (Fischetti/Lodi) or Delta/Onion Method (Guignard/Spielberg)
- Improve a given binary solution $x^*$ by restricting search space. Add inequality:

$$\sum_{j \mid x_j^* = 1} x_j + \sum_{j \mid x_j^* = 0} (1 - x_j) \leq k$$

- Solve resulting problem as regular MIP
- Solve recursively/iteratively
Computational Experiments

- 83 instances from Wolsey’s web page: http://www.core.ucl.ac.be/wolsey/ufcn.htm
- Models in Xpress-Modeler & MPS format
- Translated in consistent GAMS model with 83 data files
- Verified translated instances:
  - Same relaxed objective value
  - Matrix statistics
  - When solved optimally, same objective value
- Computing Environment
  - Athlon 2800, 2 GB memory running Linux
  - Time limit 1800 seconds
No BCH – 1800 s Time Limit
Unsolved Instances - Gap
Unsolved Instances - Objective
Unsolved Instances - Bound

![Graph showing the performance of different solvers (CPLEX, XPRESS, CPLEX+C, CPLEX+C+H) with respect to the best relative bound.](image)
Unsolved Instances - Gap
Conclusion

• BCH Facility automates all major steps necessary to define, execute, and control the use of user defined routines within the framework of general purpose MIP codes

• Cut separation models provides a new application area for Global Optimization

• More information:
  - www.gams.com/docs/bch.htm (BCH)
  - www.gams.com/apps/fcnet.htm (FC-Network)
  - www.gams.com/presentations (Presentation)