GAMS
Branch-and-Cut
& Heuristic Facility

Michael R. Bussieck MBussieck@gams.com

GAMS Software GmbH

GAMS Development Corp

INFORMS Annual Meeting

Washington DC 12 October 2008

Welcome/Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Extensions and Open Source
- Algorithm Prototyping and BCH

Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Extensions and Open Source
- Algorithm Prototyping and BCH

GAMS

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

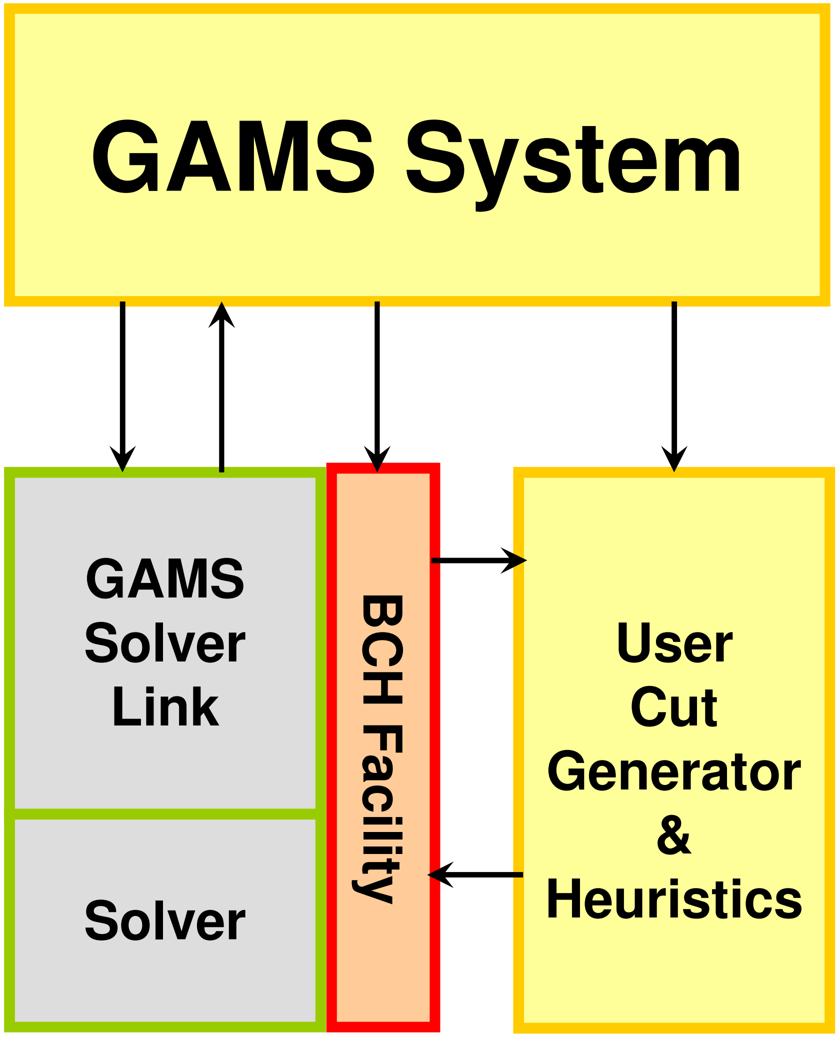
- Branch-and-Cut(-and-Price)
  - Abacus, MINTO
  - BCP, Bonmin, Cbc, SCIP, Symphony, …
  - Cplex, Xpress-MP, …

- 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
- Algorithm Prototyping and BCH

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.. z =e= sum(j, value(j)*x(j));
mk(i)..   sum(j, a(i,j)*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

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

defobj..      z =e= sum(j, (1-x.l(j))*y(j));
defcover..  sum(j, ai(j)*y(j)) =g= size_i+1;

model cover /all/; solve cover min z using mip;
```

Cover Cuts c(j)=y.l(j):
sum(c(j),x(j)) =l= card(j)-1;

Cover Cuts and Rounding Heuristic

- Activate BCH facility (option file):

```
usercutcall     mknap –goto cuts
userheurcall    mknap –goto heuristic
```

- 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);                                        // cut matrix
   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

```
        Nodes                                              Cuts/
   Node  Left     Objective  IInf  Best Integer     Best Node    ItCnt     Gap

      0     0     4134.0741     2                   4134.0741        3
*** Calling heuristic. Solution obj:   3300.0000
*     0+    0                          3300.0000    4134.0741        3   25.27%
*** Calling cut generator. Added     2 cuts
      0     0     3871.4286     2      3300.0000      User: 1        5   17.32%
*** Calling heuristic. obj = 3300
*** Calling cut generator. Added     1 cut
      0     0     3800.0000     3      3300.0000      User: 1        7   15.15%
*** Calling heuristic. obj = 3300
*** Calling cut generator. No cuts found
*** Calling cut generator. No cuts found
*** Calling heuristic. obj = 3300
      0     2     3800.0000     3      3300.0000    3800.0000        8   15.15%
*** Calling cut generator. No cuts found
*** Calling heuristic. obj = 3800
*     1     0      integral     0      3800.0000    3800.0000        9    0.00%
```

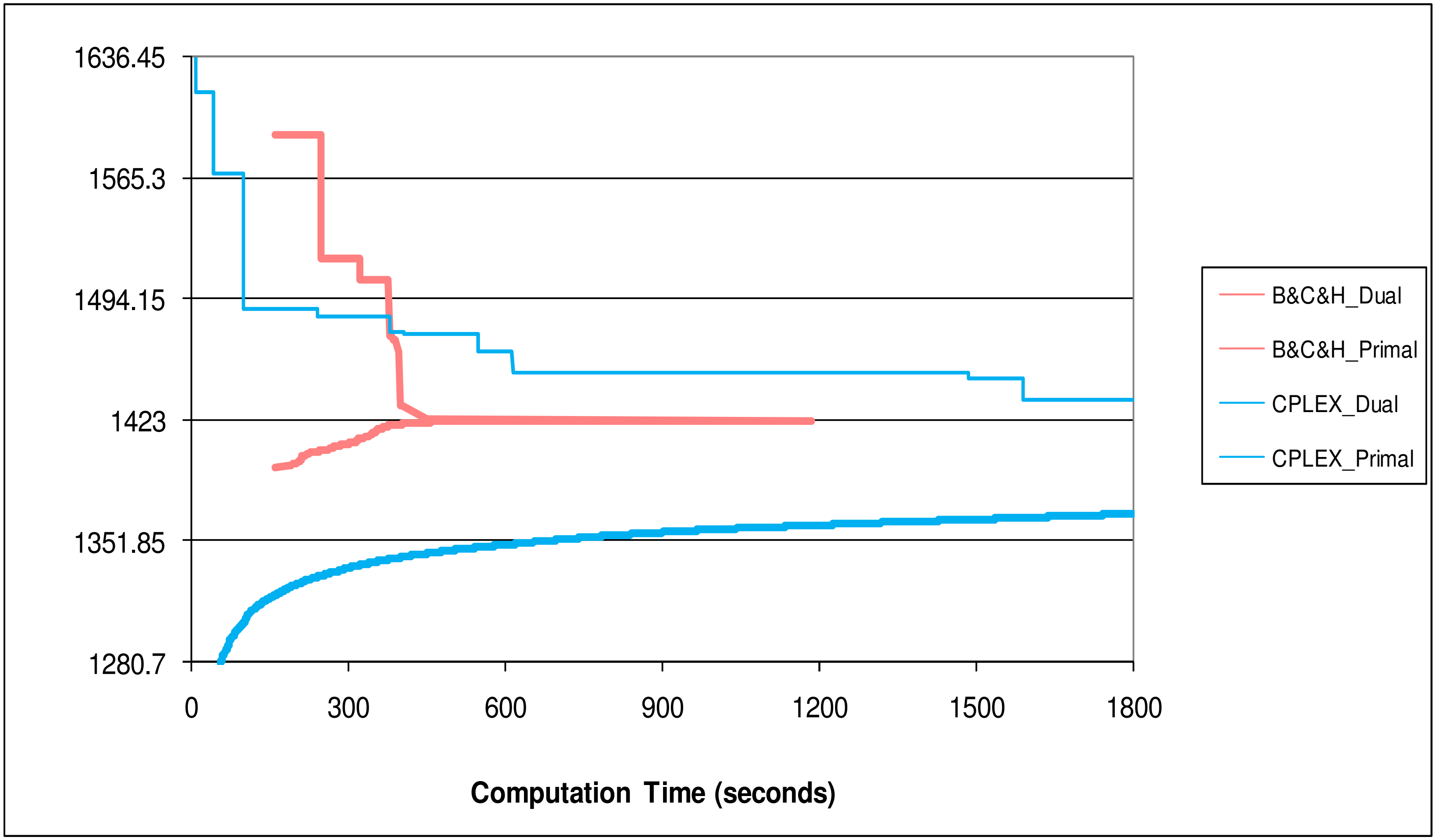
Oil Pipeline Design Problem

- Real Example: Oil Pipeline Design Problem
  - J. Brimberg, P. Hansen, K.-W. Lih, N. Mladenovic, M. Breton 2003. An Oil Pipeline Design Problem. Operations Research, Vol 51, No. 2 228-239
  - Cuts generated when new incumbent is found
  - Rounding Heuristic, Local Branching
  - http://www.gams.com/modlib/libhtml/bchoil.htm

- 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
- Algorithm Prototyping with BCH

Some Recent Extensions

- Features
  - *Cuts and Heuristics*
  - Incumbent Filters
  - Branching (Alexander Martin, TU Darmstadt)
  - Column Generation (Knut Haase, TU Dresden)
    - Pricing

- 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

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

Agenda

- Branch-and-Cut & Heuristic Facility
- First Example
- Extensions and Open Source
- Algorithm Prototyping with BCH

Quesada/Grossmann Algorithm for MINLP

- Combination of
  - Outer Approximation (OA)
  - Branch-and-Cut
- OA
  - Cycle between
    - NLP (with fixed discrete)
    - MIP master with linearizations and cuts
  - Resolve MIP master problem from scratch in every iteration
- QG
  - Keep MIP master alive and add linearizations at different points in the B&C algorithm
  - Implementations: Bonmin and FilMINT

Solve NLPR(y^l, y^u) and let (η^0, x^0, y^0) be its solution **(initialize)**
if NLPR(y^l, y^u) is infeasible **then**
  Stop. MINLP is infeasible
else
  $\mathcal{K} \leftarrow \{(x^0, y^0)\}, \mathcal{L} \leftarrow \{(y^l, y^u, \eta^0)\}, UB \leftarrow \infty, k \leftarrow 0, bk \leftarrow 0$
end if
while $\mathcal{L} \neq \emptyset$ **do**
  Select and remove node (l, u, η) from $\mathcal{L}$ **(select)**
  $\eta^k \leftarrow \eta, k \leftarrow k + 1$
  Solve CMP($\mathcal{K}, l, u$) and let $(\eta^k, \hat{x}, y^k)$ be its solution. **(evaluate)**
  if CMP($\mathcal{K}, l, u$) is infeasible OR $\eta^k \geq UB$ **then**
    Fathom node (l, u, η^k). Goto (select).
  end if
  if $y_k \in \mathbb{Z}^p$ **then**
    Solve NLP(y^k). **(update master)**
    if NLP(y^k) is feasible **then**
      $UB \leftarrow \min\{UB, z_{\text{NLP}(y^k)}\}$
      Remove all nodes in $\mathcal{L}$ whose parent objective value $\eta \geq UB$. **(fathom)**
      Let (x^k, y^k) be solution to NLP(y^k)
    else
      Let (x^k, y^k) be solution to NLPF(y^k)
    end if
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{(x^k, y^k)\}$. Goto (evaluate).
  else if Do additional linearizations **then**
    See Algorithm 3.1 **(cut)**
    Goto (evaluate)
  else
    Select b such that $y_b^k \notin \mathbb{Z}$. $bk \leftarrow k$. **(branch)**
    $\hat{u}_b \leftarrow \lfloor y_b^k \rfloor, \quad \hat{u}_j \leftarrow u_j \quad \forall j \neq b$
    $\hat{l}_b \leftarrow \lceil y_b^k \rceil, \quad \hat{l}_j \leftarrow l_j \quad \forall j \neq b$
    $\mathcal{L} \leftarrow \mathcal{L} \cup \{(l, \hat{u}, \hat{\eta}^k)\} \cup \{(\hat{l}, u, \hat{\eta}^k)\}$
  end if
end while

Simple Implementation of QG

- Start with some linearization of the problem
- Perform regular B&C
- Whenever if the MIP master finds an integer solution
  - Take the discrete variables, fix them and solve the NLP
  - If NLP is feasible and better than the incumbent
    - Install this solution as the incumbent.
    - Add the linearization at this point to the MIP master problem
  - If the NLP is infeasible, reject the solution and prevent that the solution comes up again.

- At fractional nodes, add other linearizations

BCH Implementation of QG

- GAMS/CPLEX
  - Incumbent filter callback to check if integer solution of MIP master results in a solution to MINLP
  - Cut callback to add linearizations
- Complications
  - Install MINLP solution as incumbent to MIP master problem (CPLEX).
    - Remember MINLP solution and supply this when the heuristic callback is called.
  - Pass initial linearization of MINLP to Cplex (GAMS)
  - Calculate linearization at a point (calculate derivatives) (GAMS)

Cplex Log for QG

```
Root relaxation solution time =    0.02 sec.
*** Calling cut generator. No cuts found
*** Checking incumbent with objective 8000. Rejected!
*** (QG) New incumbent 5.4299e+008 0 !!!
        Nodes                                          Cuts/
   Node  Left      Objective  IInf  Best Integer     Best Node    ItCnt     Gap

      0     0     8000.0000     0                    8000.0000        6
*** Calling cut generator. Added     1 cut
*** Calling heuristic. Solution obj: 542989682.8678
*** Checking incumbent with objective 5.4299e+008. Accepted!
*     0+    0                      5.42990e+008     8278.0645        6  100.00%
*** Calling cut generator. No cuts found
*** Calling cut generator. No cuts found
*** Checking incumbent with objective 9950. Rejected!
*** (QG) New incumbent 3.87854e+008 0 !!!
      0     0     9950.0000     0  5.42990e+008      Cuts: 15       13  100.00%
*** Calling cut generator. Added     1 cut
*** Calling cut generator. No cuts found
*** Calling heuristic. Solution obj: 387854009.1913
*** Checking incumbent with objective 3.87854e+008. Accepted!
*     0+    0                      3.87854e+008     9950.0000       13  100.00%
```

Summary

- BCH readily available with GAMS
- Implement user heuristics and cuts 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)