Column Generation in GAMS

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

Michael R. Bussieck MBussieck@gams.com

GAMS Software GmbH

GAMS Development Corp

83rd Working Group Meeting
Real World Optimization
19. – 20. November 2009

GOR

Mathematical Optimization in Transportation
- Airline, Public Transport, Railway -

Welcome/Agenda

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

Agenda

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

Modeling Systems

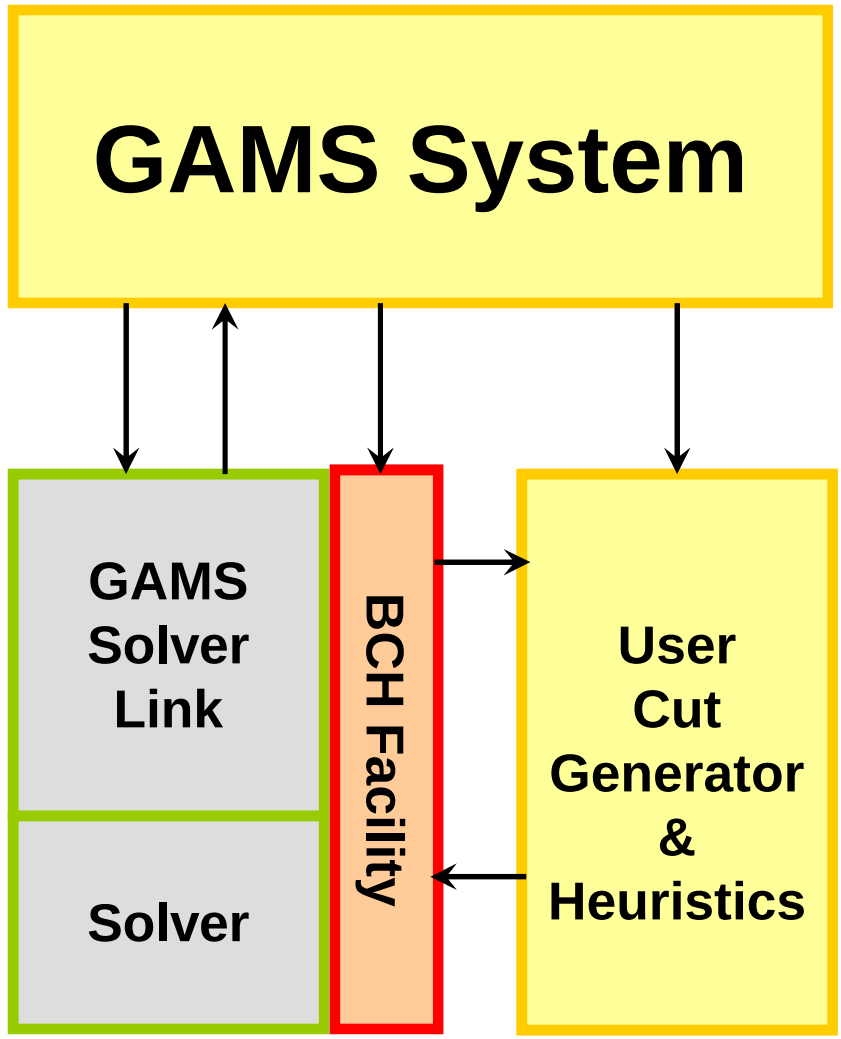
- Best way to model and solve optimization problems
- Solid foundation based on "*Separation*"
  - Separation of Model and Data
  - Separation of Model and Algorithm
- Art of Modeling
- Some Modeling Systems provide (all) features of a programming language (e.g. GAMS, MOSEL, …)
  - Avoid usual stumbling 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 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

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

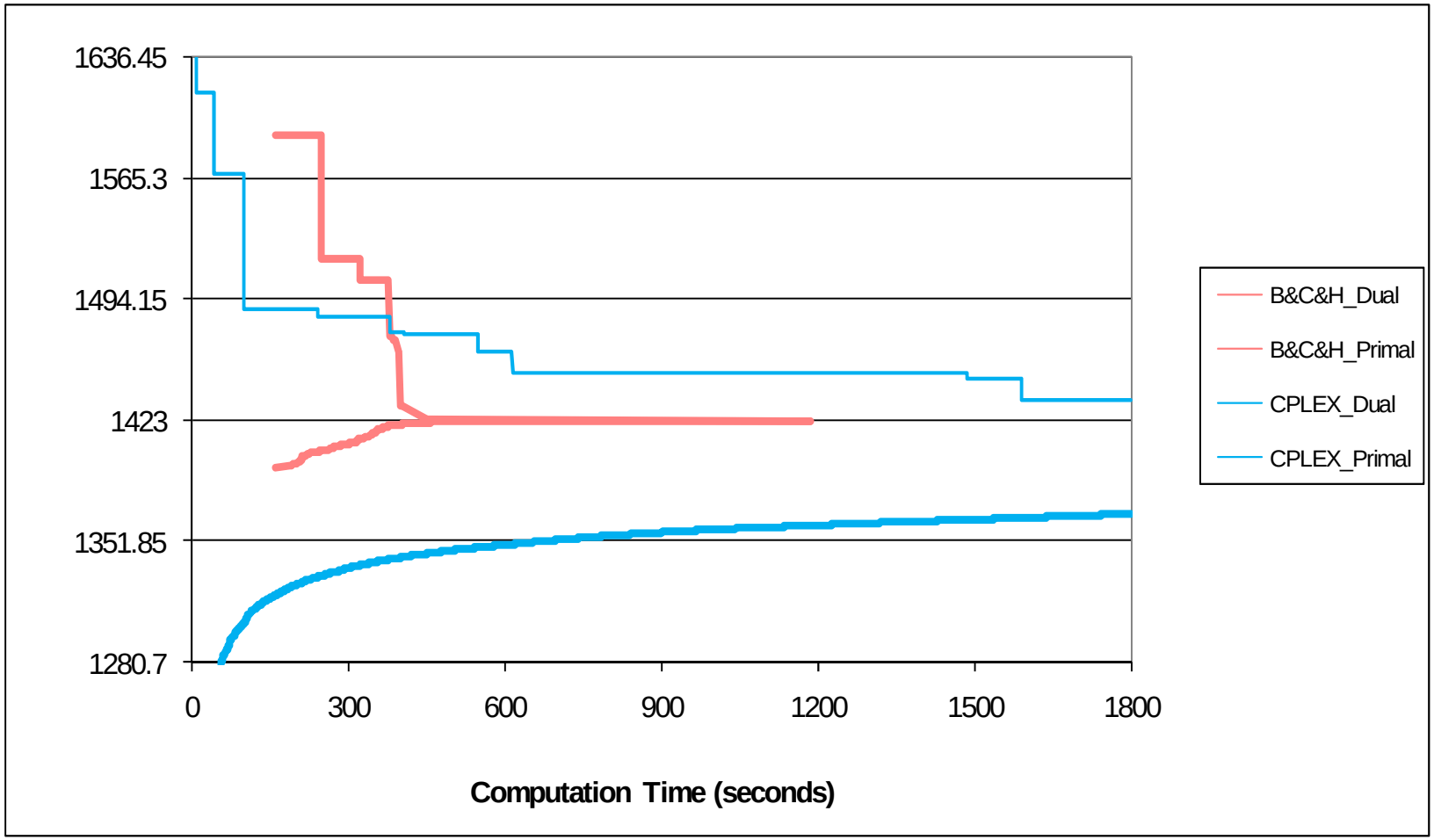
Oil Pipeline Design Problem

- 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)



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

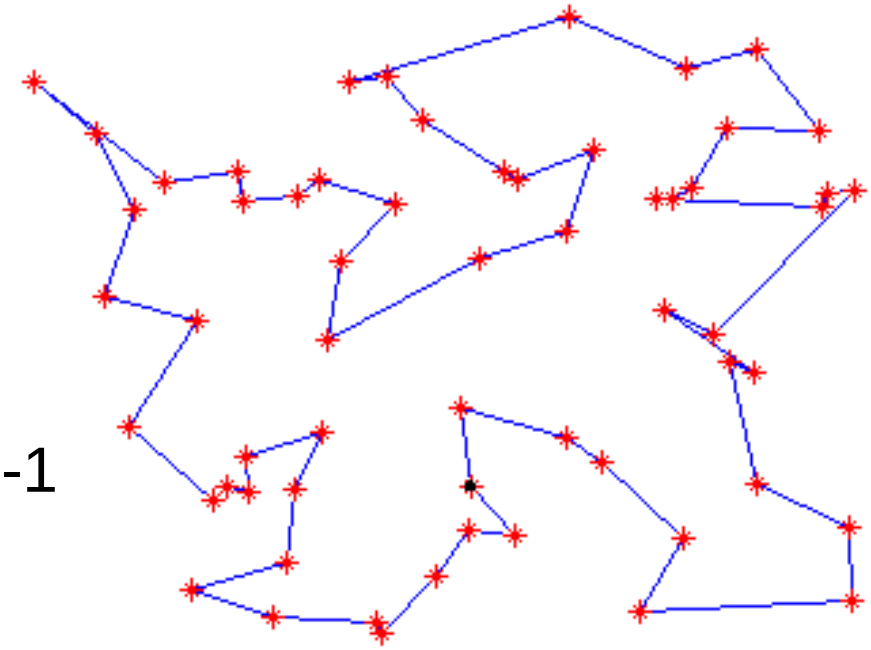
Agenda

- 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)) ≤ 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!

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

      0     0       20.0000     0                   20.0000        4
*** Calling cut generator. Added     3 cuts
*** Calling cut generator. /
*** Checking incumbent with objective 136. Rejected!
*** Calling cut generator. Added     2 cuts
*** 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

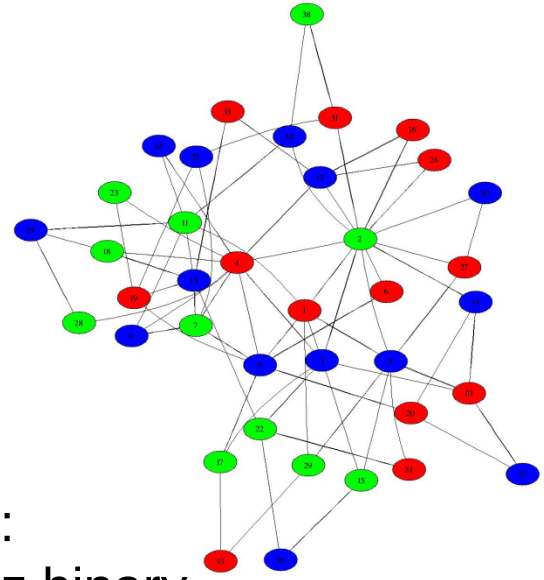
Graph Coloring Algorithm

- 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 sum(i, x(i)), st. sum(i contains v, x(i))$\geq$1,
  i independent set of (V,E)

  marginal cost ⬇

  Pricing problem:
  Find an independent set j with sum(v in j, π_v) > 1:
  max sum(v, pi(v)*z(v)), st. z(u)+z(v) $\leq$ 1, uv in E, z 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`)**

```
Iteration           Objective            In Variable           Out Variable
     1               3.252941                    x91                    x83
     2               3.252941     defcover(20) slack                    x84
LP status(1): optimal
Optimal solution found.
Objective :           3.252941
--- Calling pricing problem.
  3 columns added.
Iteration           Objective            In Variable           Out Variable
     1               3.252941                    x92     defcover(20) slack
     2               3.250000                    x80                    x78
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;
```

does not guarantee optimal solution

- 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

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)