Recent Advances in IBM ILOG CPLEX Optimization Studio

Andrea Tramontani
CPLEX Optimization, IBM
Outline

- CPLEX 12.6.1 features
  - CPLEX Python API: now includes support for Python 3
  - New parameter to control product linearization in MIQP
  - Features for distributed memory MIP

- CPLEX 12.6.1 performance improvements
  - Summary
  - Presolve

- Local implied bound cuts
  - Global vs. local implied bound cuts
  - An interesting use case
CPLEX 12.6.1 features

- CPLEX Python API and support for Python 3
  - Details in: Ryan Kersh, Wednesday 14:45 – 16:15: "Best Practices using the CPLEX Python API"

- New parameter to control product linearization in MIQP
  - Linearization of products of variables can have a large performance impact
    - E.g., MIQP can be easily transformed to a MILP if the quadratic objective function contains binary variables only
  - By default, CPLEX will adopt the strategy that looks more promising
  - The new parameter `QToLin` allows the user to explicitly disable or enable the linearization

- New features for distributed memory MIP:
  - Possibility to specify different parameter setting for each worker in distributed concurrent MIP
  - Distributed concurrent MIP and distributed parallel MIP can now be opportunistic
Outline

- CPLEX 12.6.1 features
  - CPLEX Python API: now includes support for Python 3
  - New parameter to control product linearization in MIQP
  - Features for distributed memory MIP

- CPLEX 12.6.1 performance improvements
  - Summary
  - Presolve

- Local implied bound cuts
  - Global vs. local implied bound cuts
  - An interesting use case
MIP Performance Evolution in CPLEX

Date: 5 November 2014
Testset: 3147 models (1792 in ≥ 10sec, 1554 in ≥ 100sec, 1384 in ≥ 1000sec)
Machine: Intel X5650 @ 2.67GHz, 24 GB RAM, 12 threads (deterministic since CPLEX 11.0)
Timelimit: 10,000 sec
Deterministic parallel MILP (12 threads)

CPLEX 12.6.0 vs. CPLEX 12.6.1: MIP Performance Improvement

Date: 5 November 2014
Testset: MILP: 4134 models
Machine: Intel X5650 @ 2.67GHz, 24 GB RAM, 12 threads, deterministic
Timelimit: 10,000 sec

© 2013 IBM Corporation
CPLEX 12.6.0 vs. CPLEX 12.6.1: MIP Performance Improvement (Cont. d)

Convex MIQP

Time limits: 11 / 4

251 models

>0s

1.28x

1.00

0.78

1.00

>1s

1.67x

0.60

CPLEX 12.6.0

CPLEX 12.6.1

123 models

Convex MIQCP

Time limits: 1 / 1

172 models

>0s

1.23x

1.00

0.81

1.00

>1s

1.39x

2.20

CPLEX 12.6.0

CPLEX 12.6.1

115 models

Date: 5 November 2014

Testset: Convex MIQP: 335 models, Convex MIQCP: 190 models,

Machine: Intel X5650 @ 2.67GHz, 24 GB RAM, 12 threads, deterministic

Timelimit: 10,000 sec

© 2013 IBM Corporation
The test set includes 308 QPs and 286 MIQPs
- Same algorithmic framework (spatial branch-and-bound)
- Very similar improvements on QPs and MIQPs
Performance Improvements in CPLEX 12.6.1 – Summary

- **Cuts**
  - Different separation strategies in parallel cut loop
  - Improvements in MIR cut aggregator
    - Better handling of mixed integer models with general integer variables

- **Presolve**
  - Constraint disaggregation
  - Propagation of quadratic objective function and constraints in node presolve

- **Branching**
  - Improvements in reliability branching

- **Other improvements**
  - General improvements on dynamic search (mostly for MIQP)
  - General improvements in global solver for non-convex (MI)QP
  - Improvements on QP simplex
Given a quadratic constraint with bounded variables:
\[ 0.5 x^T Q x + a^T x + b \leq 0, \]
\[ L \leq x \leq U \]

Relax the constraint in the box \([L, U]\) exposing a given variable \(x_j\):
\[ 0.5 q_j x_j^2 + c_j x_j + d \leq 0, \]
\[ L_j \leq x_j \leq U_j \]

Then compute the two roots \(R^-\) and \(R^+\) (with \(R^- < R^+\)), i.e. the solutions of
\[ 0.5 q_j x_j^2 + c_j x_j + d = 0 \]

**Case 1:** \(q_j > 0\)
- \(R^-\) and \(R^+\) are valid bounds for \(x_j\) \(\Rightarrow\) \(L_j = \max \{L_j, R^-\}, \quad U_j = \min \{U_j, R^+\}\)

**Case 2:** \(q_j < 0\)
- \(R^- < L_j\) \(\Rightarrow\) \(R^+\) is a valid **lower bound** for \(x_j\) \(\Rightarrow\) \(L_j = \max \{L_j, R^+\}\)
- \(R^+ > U_j\) \(\Rightarrow\) \(R^-\) is a valid **upper bound** for \(x_j\) \(\Rightarrow\) \(U_j = \min \{U_j, R^-\}\)
- \(R^- < L_j \leq U_j < R^+\) \(\Rightarrow\) the problem is **infeasible**
Presolve: propagation of quadratic objective and constraints (Cont. d)

- Propagation in root presolve
  - Speed-up is 0%

- Propagation in node presolve:
  - Convex MIQP
    - 16% of models affected
    - 1-4% speed-up on the affected models
  - Convex MIQCP
    - 57% of models affected
    - 12-20% speed-up on the affected models
  - Non Convex MIQP
    - 27% of models affected
    - 6-7% speed-up on the affected models
Outline

- CPLEX 12.6.1 features
  - CPLEX Python API: now includes support for Python 3
  - New parameter to control product linearization in MIQP
  - Features for distributed memory MIP

- CPLEX 12.6.1 performance improvements
  - Summary
  - Presolve

- Local implied bound cuts
  - Global vs. local implied bound cuts
  - An interesting use case
Local implied bound cuts: implications and cutting planes

- **Implications:**
  - Relationships between binary variables and non-binary variables:
    \[ z = 1 \implies y \leq D, \text{ with } z \in \{0, 1\}, \quad L \leq y \leq U, \quad D < U \]
  - Discovered during **presolve and probing**
  - Or explicitly given as **indicator constraints**

- **Imply bound cuts:**
  \[ y \leq U + (D-U) z \]

- **Obvious claim:** If \( U_1 < U_2 \), then
  \[ y \leq U_1 + (D-U_1) z \text{ dominates } y \leq U_2 + (D-U_2) z \]
Local implied bound cuts: global cuts versus local cuts

- **Global implied bound cuts:**
  - Globally valid implications
  - **Globally valid bounds** $L, U$ on implied non-binary variables
  - Separated at the root node and at branch-and-bound nodes

- **Local implied bound cuts (new in CPLEX 12.6.1):**
  - Globally valid implications
  - **Locally valid bounds** $L, U$ on implied non-binary variables
  - Separated at branch-and-bound nodes only

- **New parameter** to control local implied bound cuts:
  - -1 = disabled
  - 0 = automatic (let CPLEX choose, currently equivalent to -1)
  - 1 = moderate: separated at starts of new dives, under conservative restrictions
  - 2 = aggressive: separated at starts of new dives, more often
  - 3 = very aggressive: potentially separated at every node
Local implied bound cuts: estimated performance impact

- Enabling moderate separation of local implied bound cuts:
  - 1-2% degradation (neutral?) on our test bed

- **Key points to be investigated:**
  - Cut strength versus cut applicability
    - Local cuts are stronger, but global cuts applies to the whole tree
  - Interaction with other cut types
  - Cut filtering and cut selection
    - Local cuts are violated more often than global cuts: special filtering rules?

- **Disabled by default, but fundamental in some cases** (e.g., to handle “bigM constraints”)
Local implied bound cuts: a case study

- MIQP models arising from a classification problem (Brooks, OR 2011):

\[
\begin{align*}
\text{min} & \quad 0.5 \omega^T Q \omega + c^T y + 2c^T z \\
\text{s.t.} & \quad z_i = 0 \quad \Rightarrow \quad K_i (\omega^T a_i + \alpha) + y_i \geq 1, \quad i = 1, \ldots, n \\
& \quad z_i \in \{0, 1\}, \quad i = 1, \ldots, n \\
& \quad 0 \leq y_i \leq 2, \quad i = 1, \ldots, n \\
& \quad -M \leq \omega_j \leq M, \quad j = 1, \ldots, d \\
& \quad -M \leq \alpha \leq M.
\end{align*}
\]

with \( M = 10^8 \)

- Key point:
  - Implications are the core of the problem
  - Very large bounds on \( \alpha \) and \( \omega_j \) make any global cut completely ineffective.
Indicator constraints

\[ z_i = 0 \implies K_i (\omega^T a_i + \alpha) + y_i \geq 1 \]

are translated to

\[ K_i (\omega^T a_i + \alpha) + y_i - s_i = 1, \]
\[ s_i \geq -L_i, \]
\[ z_i = 0 \implies s_i \geq 0 \]

where \( L_i \) is inferred from global bounds on \( \alpha \) and \( \omega_j \)

Bounds \( L_i \) are **just too big** and global implied bound cuts

\[ s_i \geq -L_i z_i \]

are totally ineffective

Local implied bound cuts:

- Two levels of branching immediately yield **reasonable local bounds** on \( \alpha \) and \( \omega_j \) variables
- Local bounds \( L_i \) on \( s_i \) variables become tight
- **Local implied bound cuts are effective**
Local implied bound cuts: a case study (Cont. d, 2)

- Benchmarks on 30 models (time limit = 10k sec)
  - Solver A: CPLEX 12.6.1 default
    - 22 timeouts
    - 134.2 sec (geomean) on the 8 solved models
  - Solver B: CPLEX 12.6.1 with local implied bound cuts set to very aggressive
    - 7 timeouts
    - 469.8 sec (geomean) on the 23 solved models
    - 23.2 sec (geomean) on the 8 models also solved by default
    - Speed-up:
      - 4.8x on the 23 solved models by solver B
      - 5.8x on the 8 models solved by both solvers