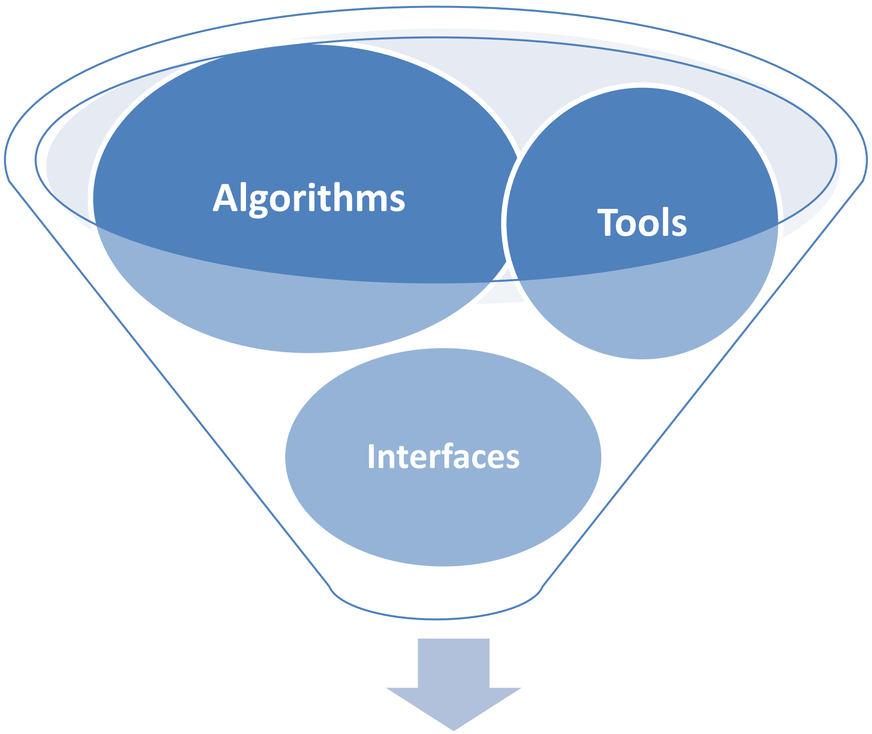
IBM ILOG CPLEX
What is inside of the box?

Ricardo Lima
rlima@andrew.cmu.edu

EWO Seminar
Carnegie Mellon University

Contents

Introduction



CPLEX

Optimization software package

Commercialized by IBM ILOG

Types of problems CPLEX can solve

Mathematical programming problems:

- Linear programming
- Mixed integer programming
- Quadratic programs
- Mixed integer quadratic programs
- Quadratic constrained programs
- Mixed integer quadratic constrained programs

- It is used to solve other problems: MINLP

Linear programming (LP)

$$\text{Maximize:} \quad c^T x$$

Objective function

$$\text{Subject to:} \quad Ax \leq b$$

Constraints

$$x \in \mathbb{R}^n$$

Decision variables

$$A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m$$

Mixed integer linear programming

Integer variables

$$
\begin{aligned}
\text{Maximize:} \quad & c^T x + \boxed{d^T y} \\
\text{Subject to:} \quad & Ax + \boxed{By} \leq b \\
& x \in \mathbb{R}^n \\
& y \in \mathbb{Z}_+^k \\
& A \in \mathbb{R}^{m \times n}, B \in \mathbb{R}^{m \times k}, c \in \mathbb{R}^n, d \in \mathbb{R}^k, b \in \mathbb{R}^m
\end{aligned}
\qquad \text{(MILP)}
$$

Quadratic programs

$$
\begin{array}{ll}
\text{Maximize:} & c^T x + 1/2 x^T Q x \\
\text{Subject to:} & Ax \leq b \\
& x \in \mathbb{R}^n \\
& A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m \\
& Q \in \mathbb{R}^{n \times n}
\end{array}
$$

(QP)

Remark: If matrix Q is positive semi-definite then the problem QP is convex.

Quadratic programs

$$
\begin{array}{ll}
\text{Maximize:} & c^T x + 1/2 x^T Q x \\
\text{Subject to:} & Ax \leq b \\
& x_l \in \mathbb{Z}_+,\ l \in N_l \\
& x_j \in \mathbb{R},\ j \in N_j \\
& A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m \\
& Q \in \mathbb{R}^{n \times n}
\end{array}
$$

(MIQP)

Remark: If matrix Q is positive semi-definite then the problem QP is convex.

Quadratic constrained programming

$$
\begin{array}{ll}
\text{Maximize:} & c^T x + 1/2 x^T Q x \\
\text{Subject to:} & Ax \leq b \\
& 1/2 x^T B_i x + a_i x \leq b_i,\ i = 1, \dots, m_1 \\
& x \in \mathbb{R}^n \\
& A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m \\
& B_i \in \mathbb{R}^{m \times n},\ i = 1, \dots, m_1
\end{array}
$$

(QCP)

Quadratic constrained programming

(MIQCP)

$$
\begin{aligned}
\text{Maximize:} \quad & c^T x + 1/2 x^T Q x \\
\text{Subject to:} \quad & Ax \leq b \\
& 1/2 x^T B_i x + a_i x \leq b_i,\ i = 1, \ldots, m_1 \\
& x_l \in \mathbb{Z}_+,\ l \in N_l \\
& x_j \in \mathbb{R},\ j \in N_j \\
& A \in \mathbb{R}^{m \times n}, c \in \mathbb{R}^n, b \in \mathbb{R}^m \\
& B_i \in \mathbb{R}^{m \times n},\ i = 1, \ldots, m_1
\end{aligned}
$$

CPLEX history and facts

Start	1988	1992	1997	2009	2010
Robert Bixby	CPLEX 1.0	CPLEX 2.0	Acquired by ILOG	Acquired by IBM	Latest stable release: 12.2
• Wrote a LP code • CPLEX Optimization Inc.	• LP optimizer	• MIP optimizer			

2008 – Bixby, Gu, and Rothberg left ILOG and found Gurobi Optimization.

CPLEX releases history

CPLEX 1.0
1988

- LP solver

CPLEX 2.0
1992

- Simple B&B
- Limited cuts

CPLEX 6.0
1998

- Simple B&B
- Limited cuts
- Simple heuristic
- *Faster dual simplex*

CPLEX releases history (cont.)

CPLEX6.5

1999

- 5 different node heuristics
- 6 types of cutting planes
  - Knapsack covers
  - GUB covers
  - Flow covers
  - Cliques
  - Implied bounds
  - Gomory mixed integer cuts

CPLEX 7.0

2000

- Semi-Continuous and Semi-Integer Variables
- Default LP method: dual simplex.
- Preprocessing
- Cuts:
  - mixed integer rounding
  - disjunctive
  - flow path

CPLEX 8.0

2002

- New Methods for Solving LP Models: Sifting
- Concurrent optimization: 1) Dual Simplex; 2) Barrier method, 3) Primal Simplex, 4) Barrier method
- New QP Capabilities
- 9 types of cutting planes

CPLEX release history (cont.)

CPLEX 9.0 2003	CPLEX 10.0 2006	CPLEX 11.0 2007	CPLEX 12.2 2010
• QCP • Relaxation Induced Neighborhood Search (RINS)	• Improvements for MIQPs • Changes in MIP start behavior • Feasible Relaxation • Indicators • Solution Polishing	• The solution pool • Tuning tool • Parallel mode	• MIP is faster • Multi-commodity flow cuts • Enhanced heuristics • Enhanced dynamic search

Computational performance

The actual computational performance is the result of a combination of different types of improvements:

LP solvers → **Cutting planes** → **Heuristics** → **Parallelization**

LP solvers
- Pre-processing
- Algebra for sparse systems
- Methods: primal, dual, barrier
- Techniques to avoid degeneracy and numerical difficulties

Cutting planes
- From theory to practice

Heuristics
- Node heuristics
- RINS
- Polishing

Parallelization
- Search in B&B
- Barrier method

Plus the machine improvements

Computational evolution for LPs

In the beginning

- 1952 - **(E48,V71) solved in 18 hours, 71 Simplex iterations.**
  Orden (1952), Hoffman et al. (1953)
- 1963 - **(E99,V77) estimated 120 man days.**
  Stigler's (1945) diet problem
- 1990 - **(E26, V71) solved in 8 hours.**
  Orchard-Hays (1990)

Evolution reported by Bixby for solving LP problems (1984:2004):

- Algorithms: Primal vs best of Primal/Dual/Barrier **3300x**
- Machines: (workstations -> PCs): **1600x**
- Net: algorithm x machine **5 300 000x**

5 days/5 300 000 = 0.08 seconds

Robert Bixby (2002), Solving real-world linear programs: a decade and more of progress. Operations research, 50(1), pp3-15

LP performance

- Computational experiments:

Size of the LP model:

# Equations	60,390
# Variables	69,582

No advanced basis was used

Results CPU (s)

	CPLEX version	
	7.1	12.2
Primal Simplex	205	45
Dual Simplex	281	51
Network Simplex	174	91
Barrier	97	**18**
Sifting	-	420

Optimization algorithms in CPLEX

Simplex optimizers	Barrier optimizer	Mixed integer optimizers
• Primal, dual, network • **LP and QP**	• **LP, QP, and QCP**	• Branch & Cut • Dynamic search • **MIP, MIQP, MIQCP**

Remarks:

- The barrier optimizer can explore the presence of multiple threads.
- The barrier optimizer cannot start from an advanced basis, and therefore it has limited application in Branch and Bound methods for MIPLs.
- Re-optimization with the simplex algorithms is faster, when starting from a previous basis.

MIP solvers in CPLEX

Mixed integer optimizers

- Branch & Cut
- **Dynamic search**
- **MIP, MIQP, MIQCP**

New algorithm to solve MIPs

- Branch & cut based
- Some user callbacks cannot be used

- IBM trade secret
- Methodology is proprietary

Examples

- POUTIL – MILP model from the GAMS library.
- RHS – MILP continuous time slot based model for scheduling of continuous processes.
- RH12 – MILP scheduling model with travelling salesman based constraints.

	POUTIL	RHS	RH12
Equations	2,178	16,886	10,421
Variables	1,260	12,156	19,134
0-1 variables	773	5,938	13,340

Computer: machine running Linux, with 8 threads Intel Xeon@ 2.66GHz

Branch and Bound (MILP)

- **Main idea**: solve MILP problems by solving a sequence of linear relaxations to provide bounds

MILP formulation

$$Z(X) = \min\{cx + fy \ : \ (x,y) \in X\}$$
where
$$X = \{(x,y) \in \mathbb{R}_+^n \times \{0,1\}^p \ : \ Ax + By \geq b\}$$

The relaxation is given by

$$Z(P_X) = \min\{cx + fy \ : \ (x,y) \in X\}$$
where
$$P_X = \{(x,y) \in \mathbb{R}_+^n \times \boxed{[0,1]^p} \ : \ Ax + By \geq b\}$$

The linear relaxation provides a lower bound on the optimal objective value:
$Z(P_X) \leq Z(X)$

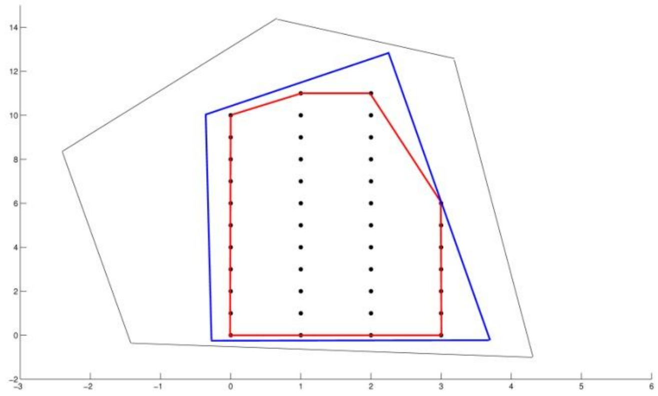
B&B algorithm

Remarks

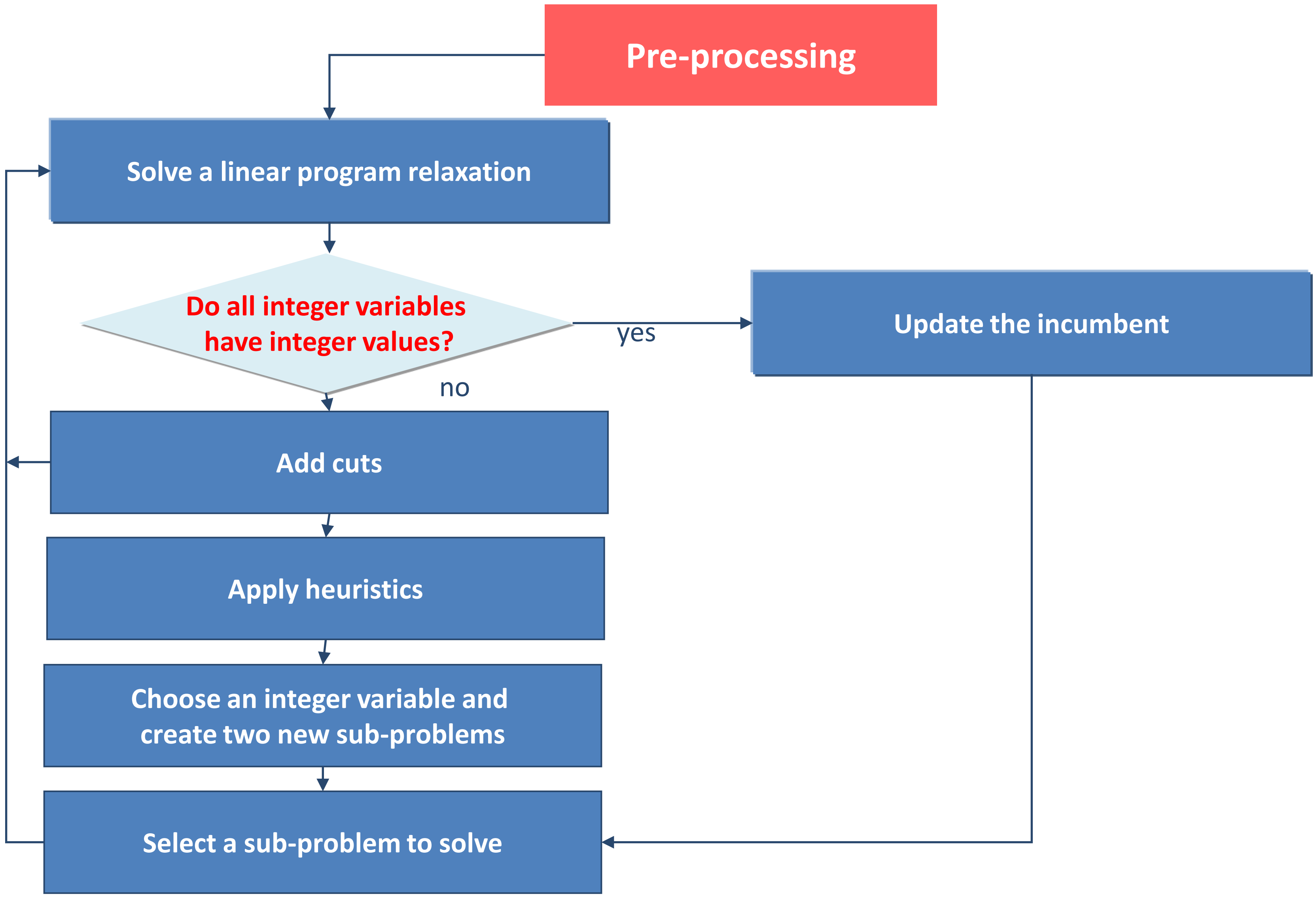
- B&B is not suitable for large scale problems
- The number of iterations grows exponentially with the number of variables

CPLEX uses the branch and cut algorithm

- Based on BB
- It is applied to a reformulation of the set V using a pre-processing step and by the addition of cutting planes.

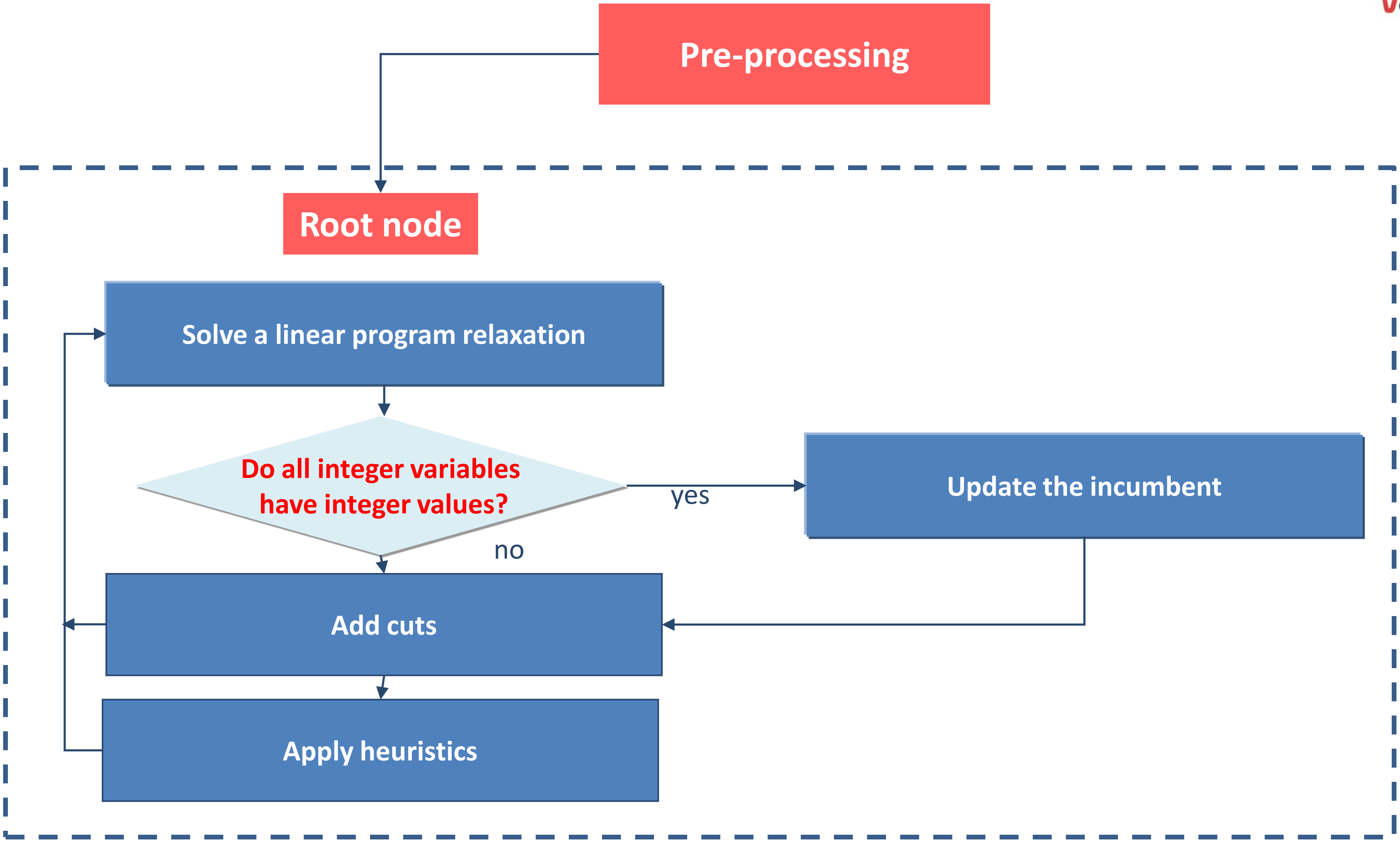


Branch and cut algorithm in CPLEX



http://www-01.ibm.com/support/docview.wss?uid=swg21400064

Branch and cut algorithm in CPLEX



http://www-01.ibm.com/support/docview.wss?uid=swg21400064

Pre-processing and probing

- **Goals**
  - Reduce the size of the problem
  - Improve the formulation
    - A new model is defined
    - Tighter formulation without increasing the size of the problem
    - Independent of the relaxation solution
- **Techniques used:**
  - Pre-processing
  - Probing

Pre-processing and probing

- **Pre-processing techniques**
  - Identification of infeasibility
  - Identification of redundancy
  - Improve bounds
  - Rounding (for MIP)

- **Probing techniques**: fix binary variables to either 0 or 1, and check the logical implications
  - Fixing variables
  - Improve coefficients
  - Logical implications

- Both formalized by Savelsbergh (1994) and Wolsey (1998)

Pre-processing example

Initial LP formulation

e1.. z =e= 2*x1 + x2 - x3;
e2.. 5*x1 -2*x2 + 8*x3 =l= 15;
e3.. 8*x1 + 3*x2 - x3 =g= 9;
e4.. x1+ x2 + x3 =l=6;
x1.up =3;
x2.up = 1;
x3.lo = 1;

Final LP formulation

--- Generating LP model P1
--- wolsey_2.gms(25) 3 Mb
--- 4 rows 4 columns 13 non-zeroes
--- Executing CPLEX: elapsed 0:00:00.017

Cplex 12.2.0.0, GAMS Link 34

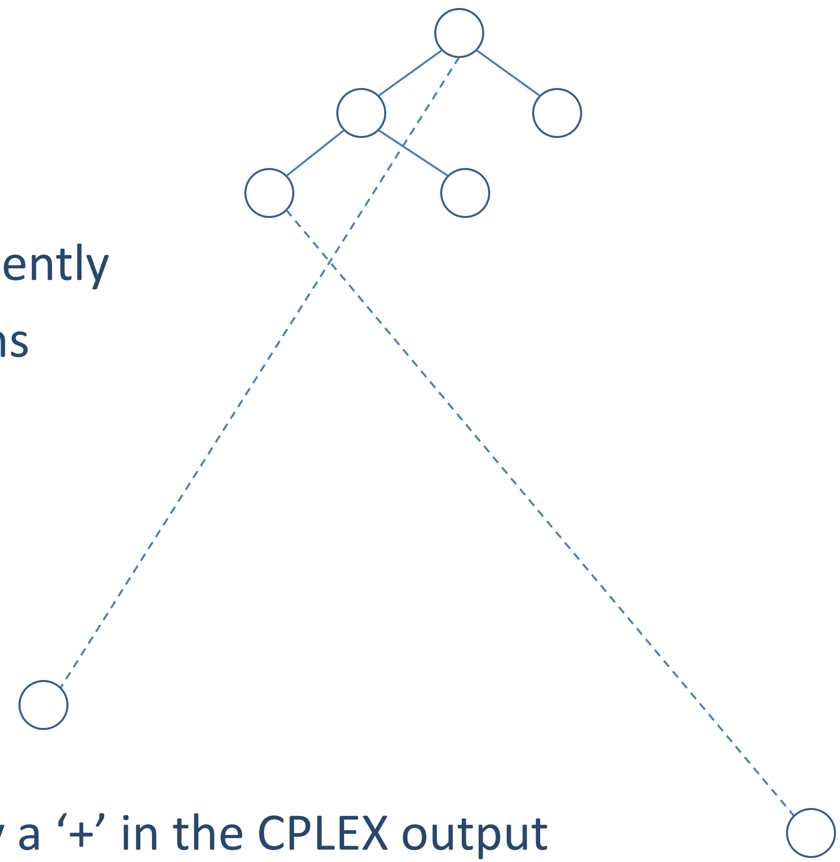
Reading data...
Starting Cplex...
Tried aggregator 1 time.
LP Presolve eliminated 4 rows and 4 columns.
All rows and columns eliminated.
Presolve time = 0.00 sec.
LP status(1): optimal

Optimal solution found.
Objective : 3.600000

Heuristics at the root node (and afterwards)

Why heuristics?

- Can achieve solutions of *difficult* MILP problems by exploring parts of the tree that the solver will not.
- May provide good solutions quickly.
- May help to prove optimality
  - explicitly: prune nodes more efficiently
  - Implicitly: provide integer solutions



Types of heuristics:

- Node heuristics: diving
- Neighborhood exploration

Note: heuristic solutions are identified by a ‘+’ in the CPLEX output

Heuristics at the root node (cont).

- Diving heuristics

  1 – Fix a set of integer infeasible variables

  2 – Bound strengthening

  3 – Solve LP relaxation

  4 - Repeat

- Neighborhood
  - Local Branching (LB)
  - *Relaxation Induced Neighborhood Search (RINS)*
  - Guided Dives (GD)
  - *Evolutionary algorithms for polishing MIP solutions*

Cuts and heuristics at the root node

- Example: MILP problem from Wolsey (1998), solved with B&C requiring 3 nodes

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

*     0+    0                          0.0000                       21     ---
      0     0      575.4371     9      0.0000       575.4371        21     ---
*     0+    0                        518.0000       575.4371        21   11.09%
      0     0      557.1433    13    518.0000        Cuts: 9        27    7.56%
*     0+    0                        525.0000       557.1433        27    6.12%
      0     0      547.8239    17    525.0000        Cuts: 9        37    4.35%
      0     0      546.4737     6    525.0000        Cuts: 8        39    4.09%
*     0+    0                        527.0000       546.4737        39    3.70%
      0     0      546.0000     6    527.0000        Cuts: 3        40    3.61%
*     0     0      integral     0    545.0000    ZeroHalf: 1        42    0.00%
      0     0        cutoff          545.0000       545.0000        42    0.00%
Elapsed real time =   0.08 sec. (tree size =  0.00 MB, solutions = 5)

Clique cuts applied:  1
Cover cuts applied:  7
Zero-half cuts applied:  8
Gomory fractional cuts applied:  1
MIP status(101): integer optimal solution
```

NEIGHBORHOOD HEURISTICS

RINS Danna, Rothberg, and Le Pape, (2005)

- **Idea:** explore the neighborhood of the incumbent to find better solutions

- **Algorithm:**
  - Fix the binary variables with the same values in the continuous relaxation and in the incumbent.
  - Solve a sub-MIP on the remaining variables.

- Example:
  - Relaxation: x=(0.1, 0, 0, 1, 0.9)
  - Incumbent: x=(1, 0, 1, 1,0)
  - Fix x_2= 0, x_4=1
  - Solve a sub-MIP

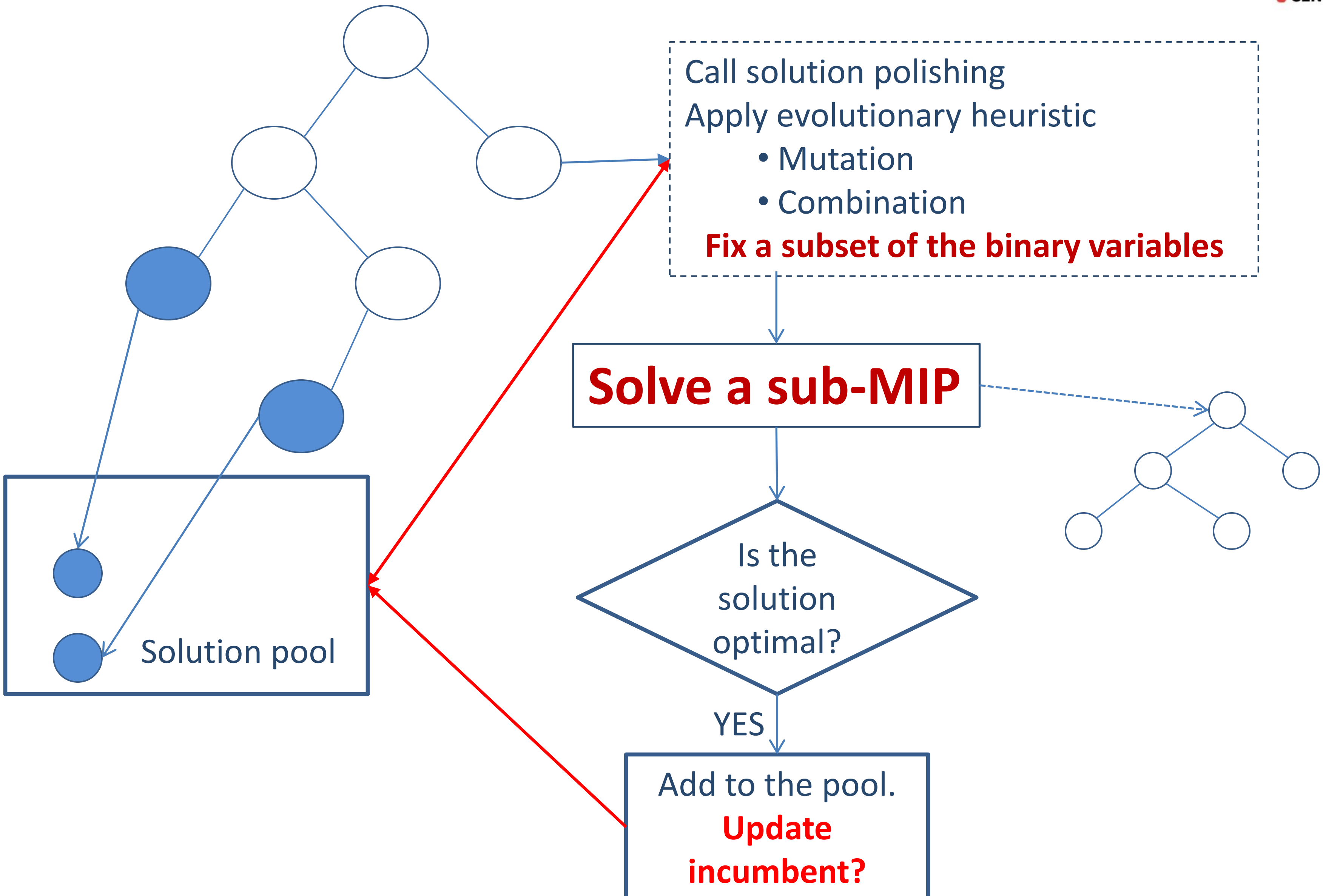
RINS (cont.)

- Remarks:
  - It may greatly improve solutions of poor quality
  - Uses the relaxation to define neighborhoods
  - Poor relaxations may lead to large sub-MIP
  - The sub-MIP are not solved optimality
  - It is only invoked every f nodes

Solution polishing Rothberg, E. (2007)

- **Idea:** explore the neighborhood of the incumbent by fixing some of the binary variables, and solving a sub-MIP.
- Polishing is based on the integration of an evolutionary algorithm ***within*** an MIP ***branch and bound*** framework.
- Can only be called when an incumbent is available.

Integration of EA and B&B



EA operators

EA steps

1. Mutation
   a) Choose a seed from the pool (random)
   b) Fix f variables (apply a random mask)
   c) Solve sub-MIP
   d) Add the solution found to the pool

Seed x=(1, 0, 0, 1, 0)

New x=(?, **0**, ?, **1**, **0**)

Solve a sub-MIP with 2 binary variables.

2. Combination
   a. Choose a pair of solutions from the pool (random)
   b. Fix variables with the same value
   c. Solve the sub-MIP
   d. Add the best solution to the pool

Seed 1 x=(1, 0, 0, 1, 0)
Seed 2 x=(0, 1, 0, 1, 0)

New x=(?, ?, **0**, **1**, **0**)

Solve a sub-MIP with 2 binary variables.

Solution polishing results Rothberg, E. (2007)

Relative gap between solution found and best known solution.
Bold means better solution.

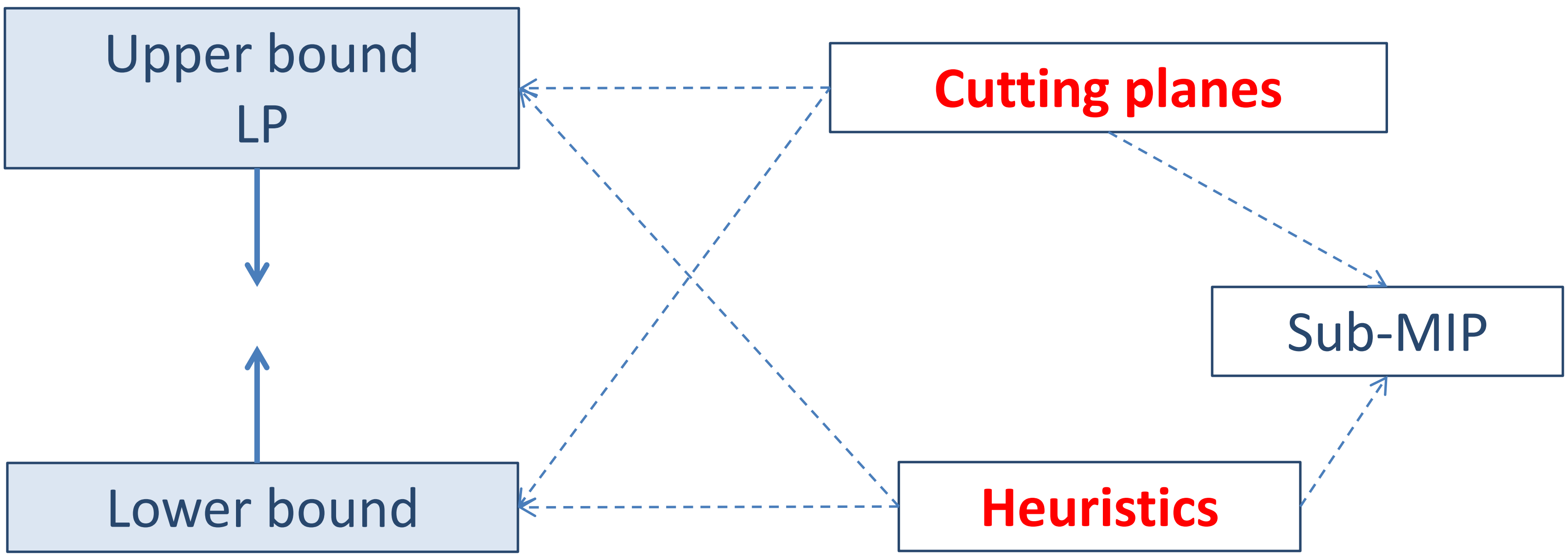
Instance	Relative solution quality (versus best known)			
		After 50% additional time		
	Initial 50K nodes GD + LB + RINS	Defaults	GD + LB + RINS	Polishing
glass4	0.34722	**0.34722**	**0.34722**	**0.34722**
liu	0.09747	0.09747	0.09567	**0.03430**
mkc	0.00020	0.00020	0.00020	**0.00000**
protfold	0.12903	0.12903	0.12903	**0.06452**
sp97ar	0.00090	0.00090	0.00081	**0.00056**
swath	0.02517	0.02517	0.02517	**0.02272**
timtab2	0.07545	0.07545	0.07545	**0.06772**
bg512142	0.04287	0.04287	0.04287	**0.00000**
dg012142	0.26215	0.26215	0.26198	**0.26137**
B2C1S1	0.00707	0.00707	0.00707	**0.00093**
pharma1	0.00288	0.00288	0.00288	**0.00129**
sp97ic	0.00360	0.00360	0.00360	**0.00000**
sp98ar	0.00083	0.00083	0.00083	**0.00079**
sp98ic	0.00289	0.00289	**0.00018**	0.00234
UMTS	0.00107	0.00107	0.00107	**0.00106**
rococoB10-011001	0.02917	0.02328	**0.00820**	0.01965
rococoB11-110001	0.03058	0.03058	**0.02938**	**0.02938**
rococoB12-111111	0.02919	0.02919	**0.00000**	0.01369
rococoC10-100001	0.06025	0.06025	**0.05911**	0.06025
rococoC11-010100	0.04050	0.01249	**0.00326**	0.04050
rococoC12-100000	0.01349	**0.01349**	**0.01349**	**0.01349**
rococoC12-111100	0.01033	0.01033	0.01033	**0.00994**
ljb2	0.01574	0.01574	0.00435	**0.00000**
ljb7	0.24904	0.24904	0.24747	**0.17195**
ljb9	0.77430	0.53763	0.57458	**0.32891**
ljb10	0.03254	0.03254	0.03254	**0.03196**
ljb12	0.32932	0.32932	0.25586	**0.18556**

Solution polishing remarks

- Requires at least one solution
- Keeps the logic of the lower and upper bound used in B&B.

- Solution polishing can be activated after:
  – Node limit
  – Time limit
  – Within a gap %

Impact of cutting planes and heuristics



Parallel optimizers in CPLEX

- Parallelization available:
  - MIP solver
  - Barrier algorithm
  - Concurrent optimization

- Concurrent optimization for solving LP and QP
  - CPLEX launches several optimizers to solve the same problem, the process terminates when the first solver stops:
    - Thread 1 - dual simplex
    - Thread 2 - barrier.
    - Thread 3 – primal simplex
    - Thread >3 - barrier run.

MIP parallel optimizer in CPLEX

- Parallelization in the B&B
  - Solution of the root node
  - Solution of nodes
  - Strong branching in parallel

- 2 modes are available:
  - Deterministic – invariance and repeatability of the search path and results
  - Opportunistic – each run may lead to a different search path and results – usually out-performs the deterministic

Which one should be used?

Log of the parallelization

- **Deterministic**

```
Root node processing (before b&c):
  Real time             =   37.31
Parallel b&c, 8 threads:
  Real time             = 3565.95
  Sync time (average)   =   93.98
  Wait time (average)   =  216.70
                          -------
Total (root+branch&cut) = 3603.26 sec.
```

- **Opportunistic**

```
Root node processing (before b&c):
  Real time             =   34.47
Parallel b&c, 8 threads:
  Real time             = 3566.18
  Sync time (average)   =    5.97
  Wait time (average)   =    4.76
                          -------
Total (root+branch&cut) = 3600.65 sec.
```

Example: POUTIL

RMIP root 246,984.7

CPLEX 12.2

			Objective function	
Threads	CPU time (s)	Gap (%)	RMIP	MIP
1	**950**	0.0	266,793.0	266,793.0
4D	**211**	0.0	266,793.0	266,793.0

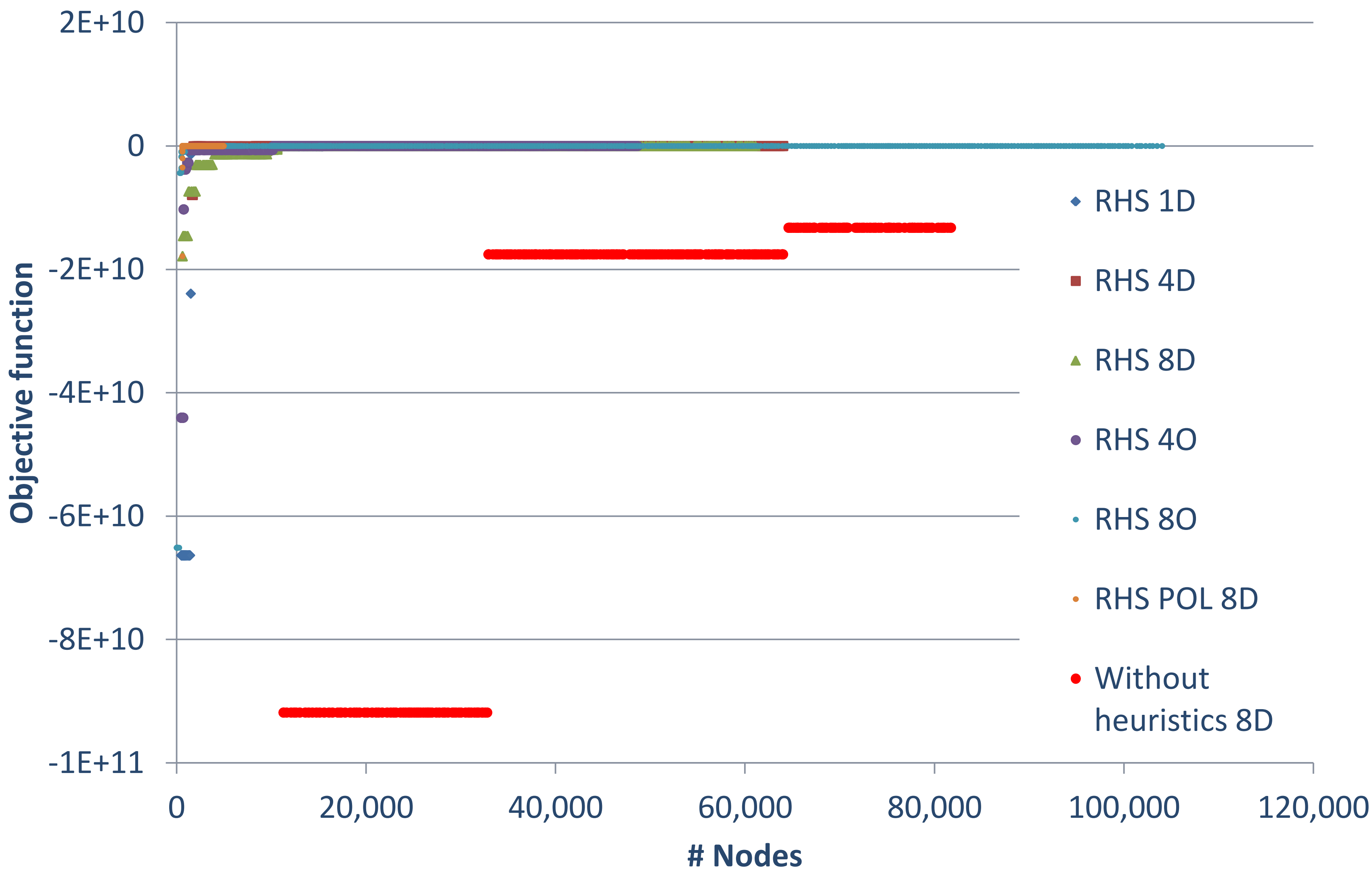
Example RH12

RMIP root 5,225,207

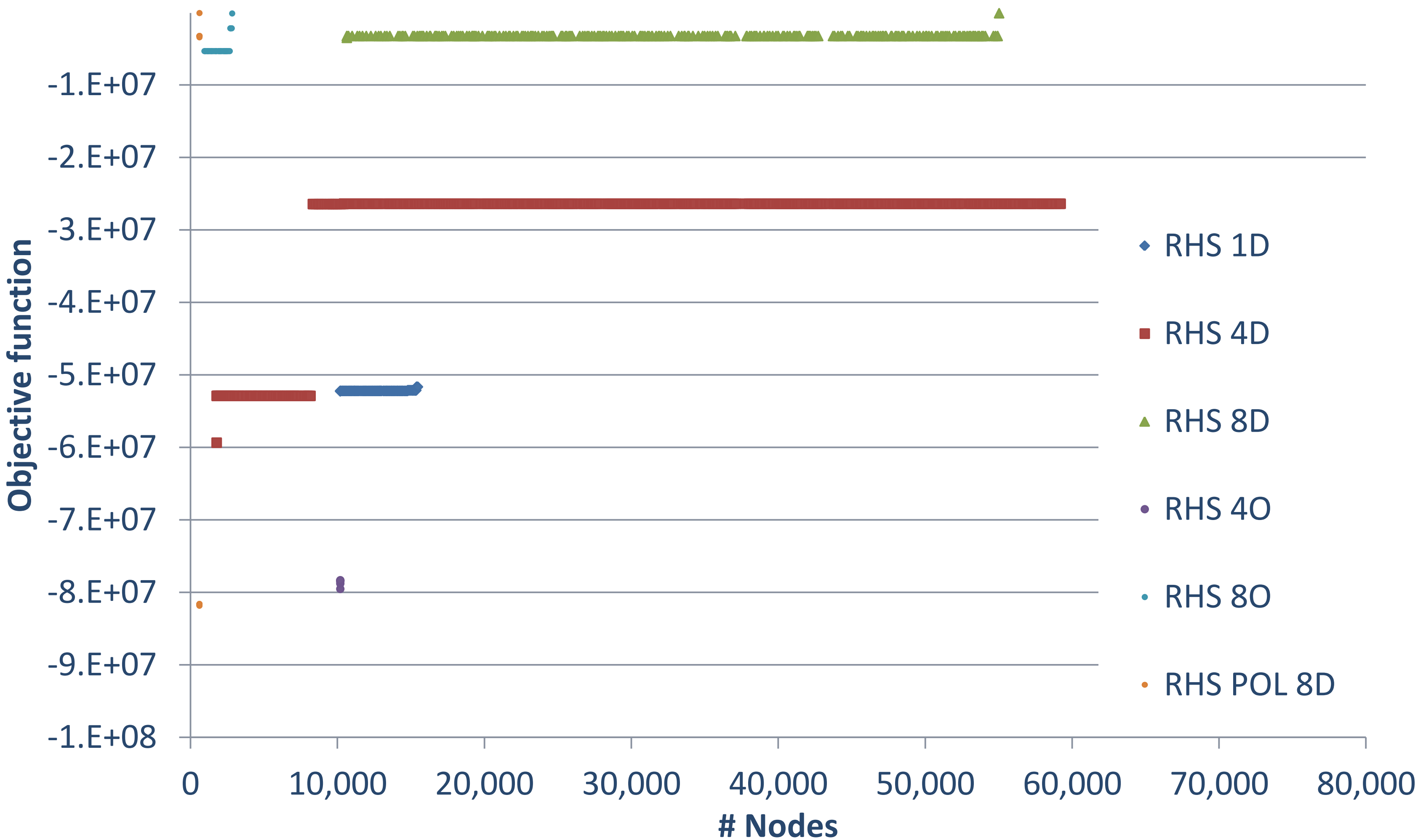
CPLEX 12.0

Threads	CPU time (s)	Gap (%)	Objective function MIP	MIP
1	3,600	101.2	5,166,820	**-444,529,600**
4D	3,600	114.8	5,165,611	-34,831,279
4O	3,600	10.5	5,166,242	**4,674,076**
8D	3,600	42	5,166,870	3,639,156
8O - 1st run	3,600	1124.5	5,165,035	-**504,162**
8O - 2nd run	3,600	17.1	5,168,434	**4,412,006**

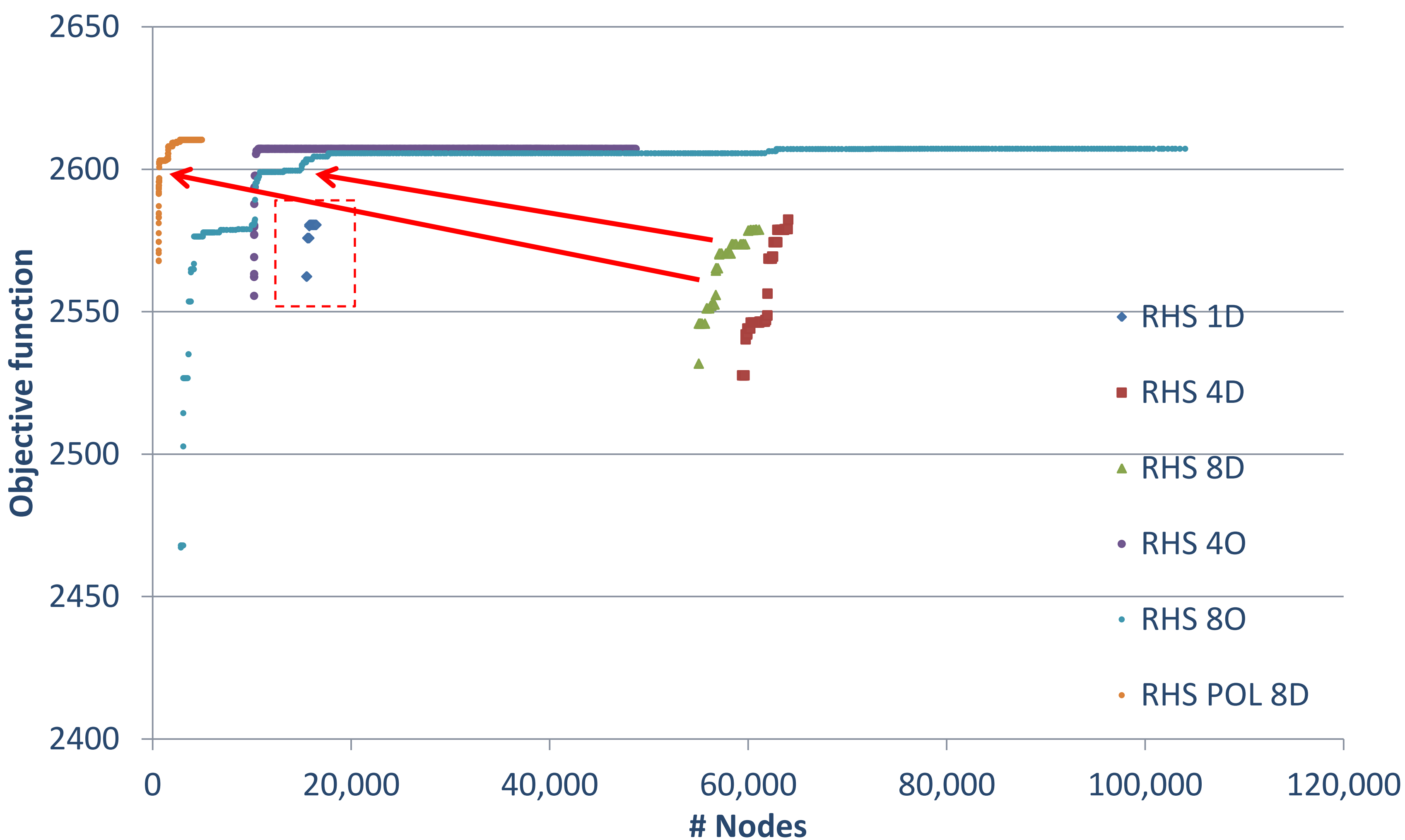
Effect of parallelization and polishing



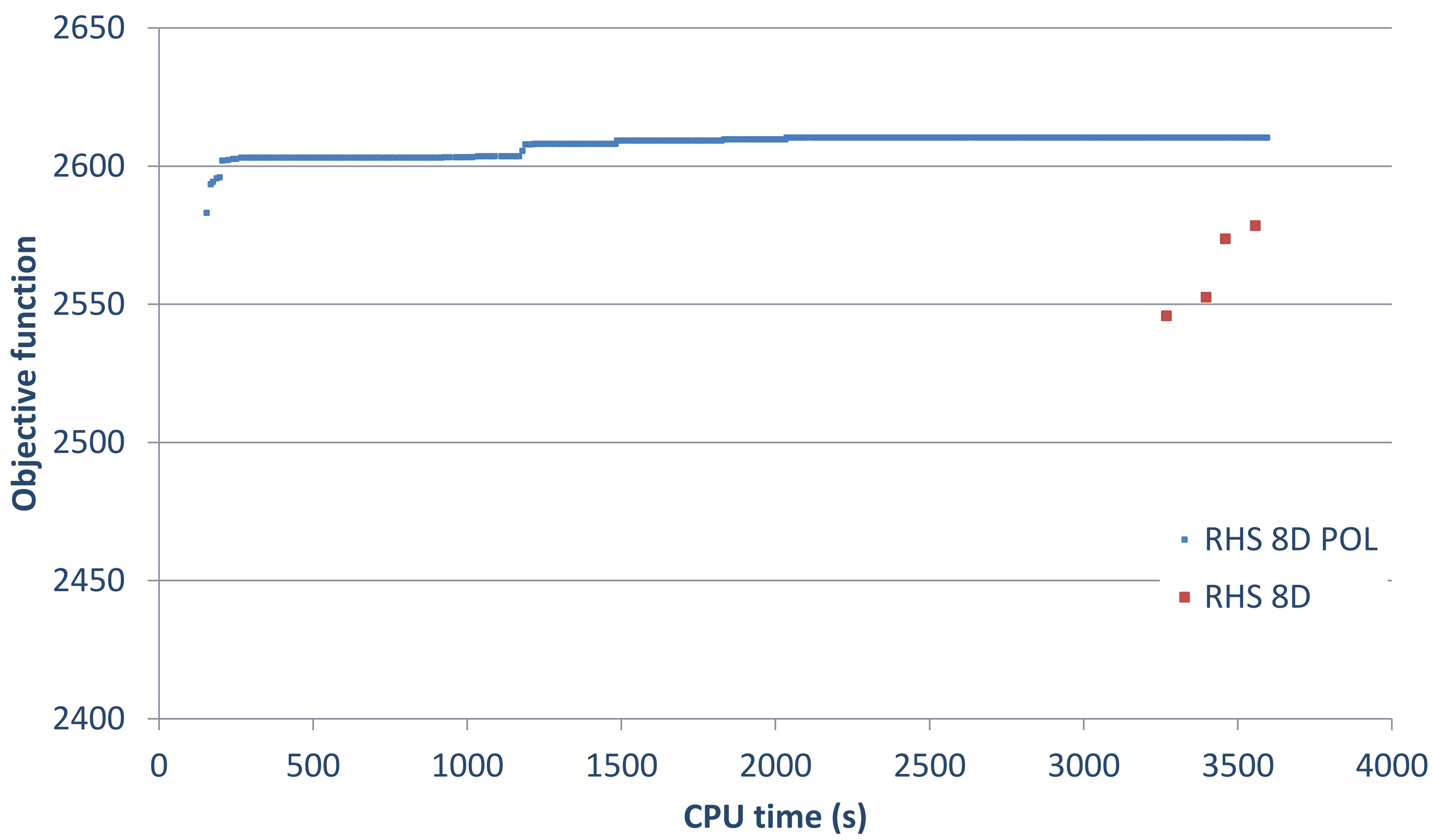
Effect of parallelization and polishing



Effect of parallelization and polishing



Impact of the solution polish option



Ineffective solution polishing

RMIP root 246,984.7

CPLEX 12.2

Threads	CPU time (s)	Gap (%)	Objective function RMIP	Objective function MIP
1	**950**	0.0	266,793.0	266,793.0
4D	**211**	0.0	266,793.0	266,793.0
4O	**206**	0.0	266,793.0	266,793.0
8D	**95**	0.0	266,793.0	266,793.0
8O	**61**	0.0	266,793.0	266,793.0
8D Polishing	**1000**	**0.94**	**264,291.7**	**266,793.0**

MIP start

- CPLEX has the option to start from a user-defined solution
  - The solution can be feasible or unfeasible
  - If the solution is not feasible, CPLEX uses a heuristic to try to repair the solution
    - Helps to find a feasible solution
  - If the solution is feasible, heuristics such as RINS or solution polishing can be used
  - Useful to debug a model

Integration of MIP start and polishing

Solve MILP
Max time: 300s
Solution Polishing after 60s
Turn off cuts

← Avoid time generating cuts

↓

Feed solution to solver
Turn on cuts

↓

Solve MILP
Solution polishing to improve solution > 90% time

← Restart, and use a starting solution to improve the search.

RHS results: polishing and MIP start

CPLEX 12.2

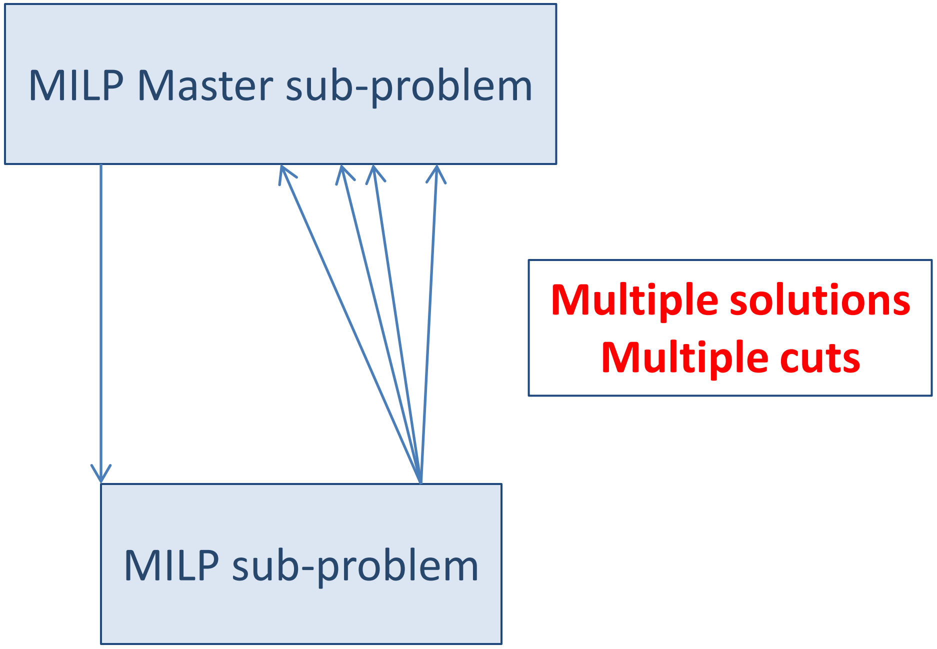
			Objective function	
Threads	CPU time (s)	Gap (%)	RMIP	MIP
1	3600	3.4	2,669.0	2,580.5
4D	3600	3.3	2,667.5	2,582.4
4O	3600	2.3	2,667.2	2,607.2
8D	3600	3.4	**2,666.3**	**2,578.8**
8O	3600	2.3	2,665.9	2,607.2
8D P - 60s	3600	2.2	2,668.8	**2,610.4**
8D Start	3600	**2.0**	**2,656.6**	2,603.5
CPLEX 7.1	3600	-	2,687.9	-

Solution pools

- Motivation:
  - Value on having more than one solution
  - Model does not capture the full essence of the process
  - Approximations on creating the model
  - Data is not accurate
- Goal: generate and keep multiple solution
  - MIP, MIQCP
- Options and tools:
  - Collect solutions with a given percentage of the optimal solution
  - Collect diverse solutions
  - Collect solutions with diverse properties
  - Difficult to implement with rolling horizon decompositions

Solution pools (cont.)

- Example of application (Emilie Danna, CPLEX)



Remark: difficult to implement with rolling horizon decompositions

Tuning tool

- Motivation
  - MIP solvers have multiple algorithm parameters
  - The performance of the solver depends on these parameters
  - Default values in solvers are defined in order to work well for a large collection of problems
    - May not work for the user specific problem

- Goal: identify the solver parameters that improve the performance of the solver for a given set of problems.

Tuning tool: example

CPLEX 12.2

Threads	CPU time (s)	Gap (%)	Objective function RMIP	MIP
1	**949**	0.0	266,793.0	266,793.0
8D	**95**	0.0	266,793.0	266,793.0

Apply the tuning tool
Time = 327s

threads 8
cutpass=-1
heurfreq=-1
itlim=100000000
parallelmode=1
probe=-1
varsel=4

CPLEX 12.2

Threads	CPU time (s)	Gap (%)	Objective function RMIP	MIP
1	**67**	0.0	266,793.0	266,793.0
8D	**8**	0.0	266,793.0	266,793.0

Variability

- **Variability** in the performance may occur in CPLEX 12.2 due to
  - Opportunistic parallelization
  - Heuristics: polishing option (random seed)
  - Numerical reasons

- Variability may occur on
  - Computational time
  - Performance in terms of nodes, iterations
  - Quality of the solution

Remarks:

- It seems particularly relevant when optimality cannot be guaranteed within the maximum time set.

- If repeatability of the results is required the above options should not be used, mainly in the development phase.

- However, it is an opportunity to obtain better solutions.

Final remarks

- The increasing performance of CPLEX has been allowing us to solve more complex problems.
- The CPLEX default parameters may not be a good choice for all problems.
- The solution pool may be an important feature to implement some decompositions.
- Topics not discussed:
  - Infeasibility analysis tool
  - Interface of CPLEX with other applications and programming languages
  - Comparison of the CPLEX performance with other solvers
  - Use of callbacks

CPLEX performance tuning (by Ricardo Lima)

- Technical support from IBM ILOG: “CPLEX Performance Tuning for Mixed Integer Programs”
  - http://www-01.ibm.com/support/docview.wss?uid=swg21400023
- Approach to tune CPLEX for MILPs

1. Use a good formulation.
2. Solve with default values.
3. Check the CPLEX log to evaluate:
   a) if it is difficult to find the first integer solution.
   b) the progress of the lower and upper bound, and determine if it is difficult to obtain integer solutions.
4. Diversify or change the search path:
   a) Set priorities for the variables.
   b) Increase the frequency of the use of heuristics if it is difficult to find integer solutions.
   c) Use the polishing option to improve the incumbent. When the polishing option is activated, CPLEX will spend more time solving sub-MIPs, and little progress is made on the relaxation.
   d) Use the parallel mode with the opportunistic option.
   e) Change the branching strategy
5. Improve the linear relaxation solution
   a) Increase the level of generation of cuts (increases the computational times)
   b) Increase the level of probing (increases the computational times)
6. If the goal is to decrease the computational time, turn off heuristics and turn off the generation of cutting planes, it may be faster.
7. Use the tuning tool.

References for CPLEX and MIP

- **CPLEX manuals**
  - IBM ILOG CPLEX Manual
    - http://publib.boulder.ibm.com/infocenter/cosinfoc/v12r2/topic/ilog.odms.cplex.help/Content/Optimization/Documentation/CPLEX/_pubskel/CPLEX.html
- **Presolve and conflict analysis**
  - Rothberg, E., ILOG, Inc. The CPLEX Library: Presolve and Cutting Planes
  - Linderoth, J. (2004). Preprocessing and Probing for integer programs, DIMACS Reconnect Conference on MIP.
  - Savelsbergh M.W.P. (1994). Preprocessing and probing techniques for Mixed Integer Programming problems. *ORSA Journal on Computing*, 6(4), p. 445-454.
  - Atamurk, A., Nemhauser, G., Savelsbergh, M.W.P., (2000). Conflict graphs in solving integer programming problems. *European Journal of Operational Research*, 121, p. 40-55.

References (cont.)

- **Branch and bound and LP**
  - Land A. H., Doig, A. G. (1960), an automatic method for solving discrete programming problems, *Econometrica*, 28, pp 497-520
  - Rothberg E., ILOG, Inc. The CPLEX Library: Mixed Integer Programming
  - Rothberg, E., ILOG, Inc. The CPLEX Library: Presolve and Cutting Planes
  - Wolsey, L. A., (1998), Integer programming, Wiley-Intersience.
- **Local search heuristics**
  - Rothberg, E. ILOG, Inc. The CPLEX Library: MIP Heuristics
  - Danna, E., Rothberg, E., Le Pape, C., (2005). Exploring relaxation induced neighborhoods to improve MIP solutions, *Mathematical Programming*, 102(1), p. 71-91.
  - Rothberg, E. (2007). An evolutionary algorithm for polishing Mixed Integer Programming Solutions. *INFORMS Journal On Computing*, 19(4) p. 534-541.
  - Fischetti, M., Lodi, A. (2005). Local branching. Mathematical Programming, 98, p. 23-47.

References (cont.)

- **Local search heuristics (cont.)**
  - Chinneck, J. and Lodi, A., (2010). Heuristics for feasibility and optimality in mixed integer programming. CIRRELT Spring School on Logistics, Montreal.
  - Dana E. (2008). Performance variability in mixed integer programming. MIP 2008
- **Parallelization**
  - Crainic, T. G., Cun, B., Roucairel, C., (2006). Parallel branch-and-bound algorithms, Parallel combinatorial optimization, Chap. 1. John Wiley and Sons, NJ.

EXTRA SLIDES

Other software packages

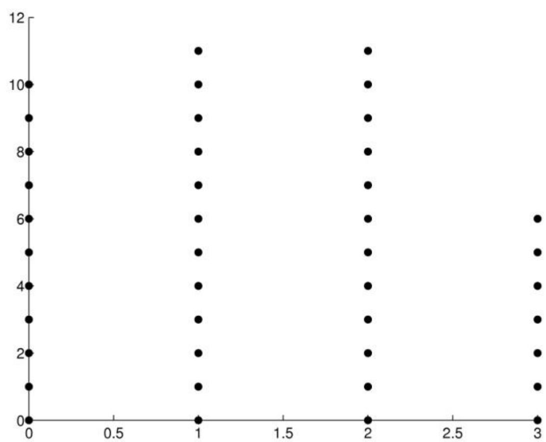
- Commercial
  - XPRESS, FICO
  - XA, Sunset Software Technology
  - MOSEK, MOSEK
  - GUROBI, GUROBI Optimization
- Non-commercial
  - SCIP, ZIB
  - MINTO,CORAL
  - GLPK, GNU
  - CBC, COIN-OR
  - SYMPHONY, COIN_OR
- Benchmark sites:
  - http://miplib.zib.de
  - http://plato.asu.edu/ftp/milpc.html

Example

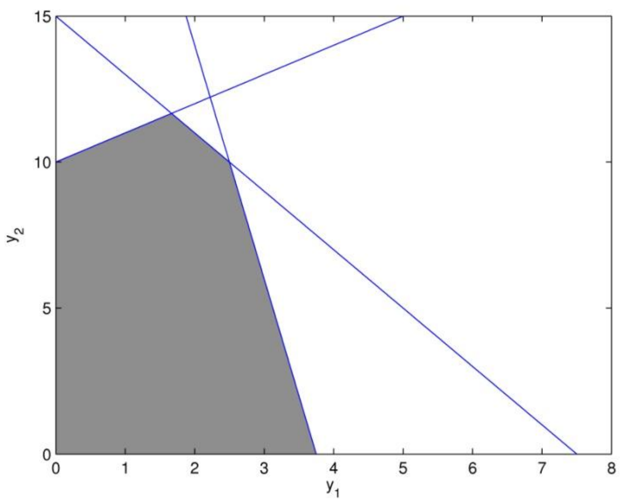
- Consider the pure integer programming problem:

$$\begin{aligned}
&\min z = -5y_1 - 2y_2 \\
&st. \\
&-y_1 + y_2 \leq 10 \\
&2y_1 + y_2 \leq 15 \\
&8y_1 + y_2 \leq 30 \\
&y_1, y_2 \in \mathbb{Z}_+
\end{aligned}$$

Feasible space



Relaxation of the feasible space



Divide et impera

Initialization

$\underline{L} = \{P_X\}$

$\overline{Z} := +\infty$

Branching

when $Z(V) \le \overline{Z}$ and $y_j^V \notin \mathbb{Z}$

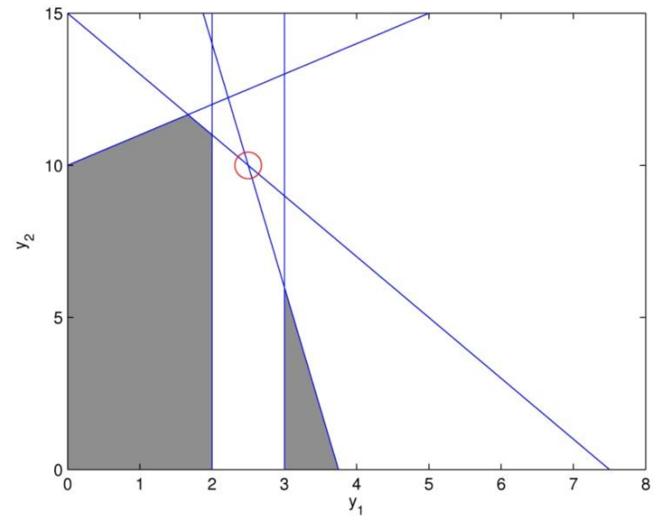
select branching variable $y_j^V \notin \mathbb{Z}$

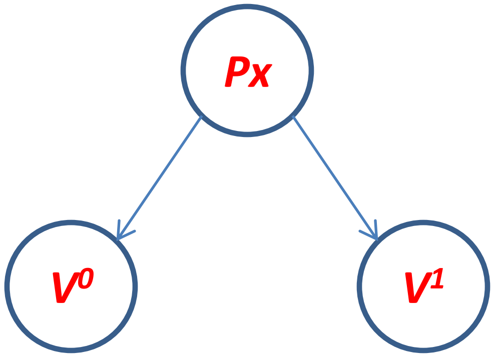
set $L := L \cup \{V^0, V^1\}$ where

$V^0 = V \cap \{(x,y) \in \mathbb{R}_+^n \times \mathbb{R}_+^p \; : \; y_j \le \lfloor y_j^V \rfloor\}$

$V^1 = V \cap \{(x,y) \in \mathbb{R}_+^n \times \mathbb{R}_+^p \; : \; y_j \le \lceil y_j^V \rceil\}$

GO TO Termination





$$\begin{aligned} &\min z = -5y_1 - 2y_2 \\ &st. \\ &-y_1 + y_2 \le 10 \\ &2y_1 + y_2 \le 15 \\ &8y_1 + y_2 \le 30 \\ &y_1 \le 2 \\ &y_1, y_2 \in \mathbb{Z}_+ \end{aligned}$$

$$\begin{aligned} &\min z = -5y_1 - 2y_2 \\ &st. \\ &-y_1 + y_2 \le 10 \\ &2y_1 + y_2 \le 15 \\ &8y_1 + y_2 \le 30 \\ &y_1 \ge 3 \\ &y_1, y_2 \in \mathbb{Z}_+ \end{aligned}$$

Divide et impera

Initialization

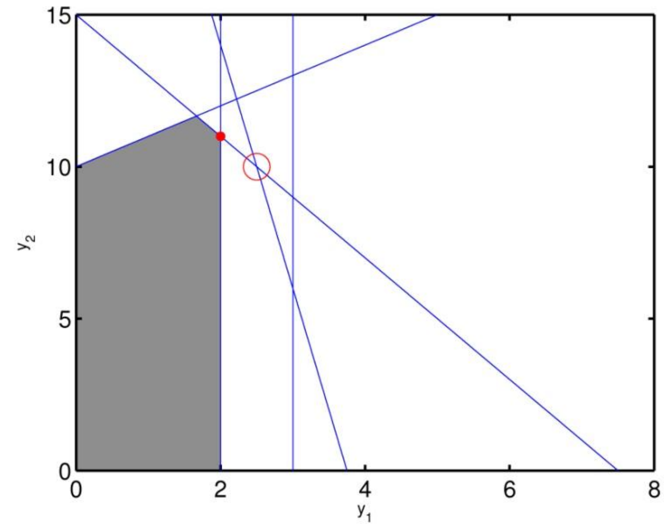
$\underline{L} = \{P_X\}$

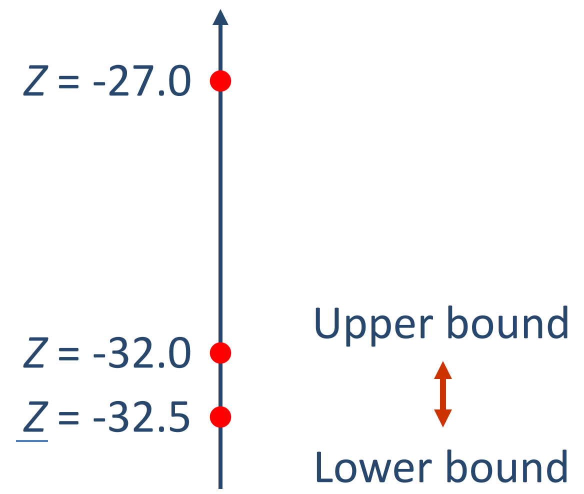
$\overline{Z} := +\infty$

Node selection and solve

Select $V \in L$ and let $L := L \setminus \{V\}$

Compute $Z(V)$, (x^V, y^V)





Cuts and heuristics at the root node

- Given: is a vector of variables $x \in \{0,1\}^p$ that by optimality can be treated as continuous, to $x \in [0,1]^p$.

- Question: what is the impact of relaxing the variables? (number of variables, relaxation, search)

Example

In the RHS model the binary variables $Z_{i,l,m,t}$ and $TRT_{i,k,m,t}$ can be relaxed to continuous variables

Reduction of the number of binary variables: 5581 to 1502.

LP solution and relaxation

- LP solution is the same for both models

  Optimal solution found.

  Objective : 2692.510176

- However, the LP relaxation is different at the beginning of the root node iterations.

CPLEX log using *Z* and *TRT* as continuous variables

```
      0     0     2690.3084  1001                    2690.3084      9175
      0     0     2688.2465   897                    Cuts: 286     11483
      0     0     2687.0382   906                    Cuts: 202     13859
      0     0     2686.7985   863                     Cuts: 97     14924
      0     0     2686.6539   881                     Cuts: 56     15602
      0     0     2686.5623   885                     Cuts: 40     15957
      0     0     2686.5612   863                  Flowcuts: 9     16028
      0     0     2686.5612   866                     Cuts: 17     16073
Heuristic still looking.
      0     2     2686.5612   866                    2686.5612     16073
Elapsed real time =  24.64 sec. (tree size =  0.01 MB, solutions = 0)
 75029 58991     2652.7284   501     2595.3987     2680.4322 24025154    3.28%
```

LP solution and relaxation

CPLEX log using Z and TRT as binary variables

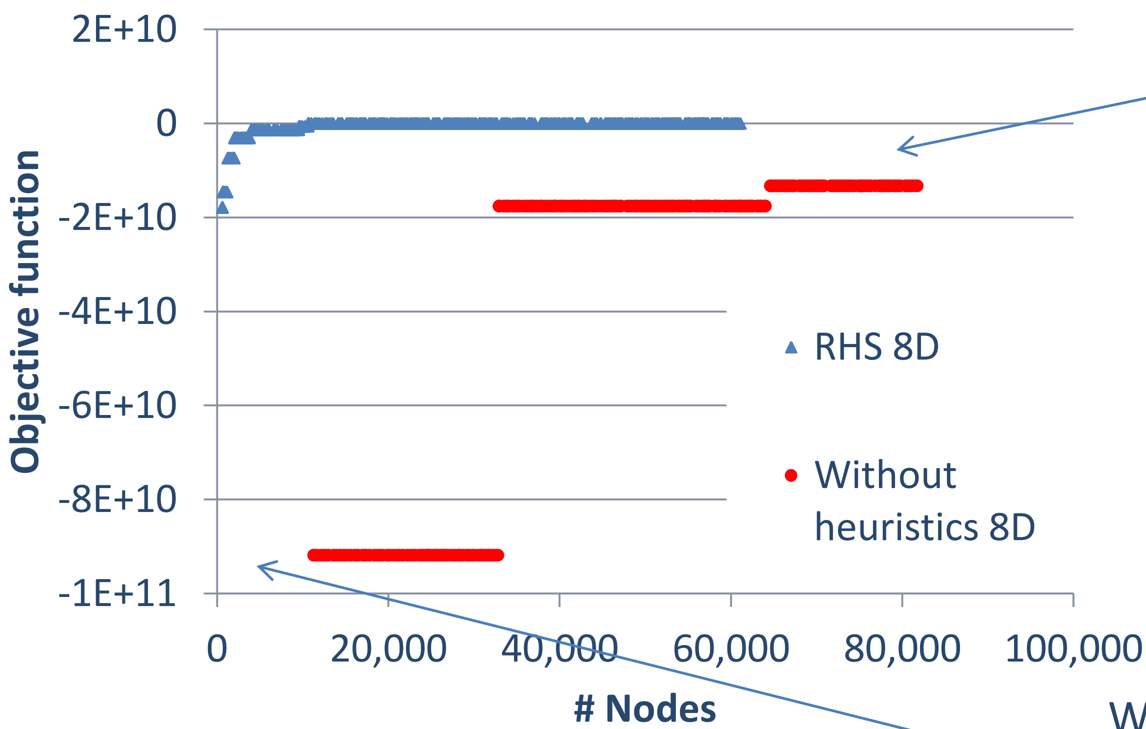
```
     0      0     2692.1693   1661                     2692.1693     11471
     0      0     2689.1996   1511                     Cuts: 365     14327
     0      0     2684.7527   1567                     Cuts: 378     16553
     0      0     2683.4370   1490                     Cuts: 263     19210
     0      0     2682.3135   1484                     Cuts: 169     20982
     0      0     2681.2411   1595                     Cuts: 143     22425
     0      0     2680.6783   1510                     Cuts: 134     24554
     0      0     2679.2076   1467                     Cuts: 119     26157
     0      0                                                        28551
     0      0                                                        29187
     0      0                                                        31526
     0      0                                                        32456
     0      0                                                        32775
     0      0                                                        33240
     0      0                                                        34183
     0      2                                                        34183
Elapsed real tim                                                 ; = 0)
  61105 47491                                                 )510359      3.39%
```

	RMIP	RMIP root Beginning	RMIP root End	RMIP Final
BIN	2,693	2,692	2,676	**2,666**
CONT	2,693	2,690	2,690	**2,680**

- The initial LP relaxations at the root node are different
- The solutions at the end of the root node are different: **2686.5612 vs 2676.4693**
- The final relaxation is better when using binary variables

Heuristics motivational example

- RHS problem optimized with heuristics and heuristics turned off.



Without heuristics the solution is not close to the best know solution.

Without heuristics a feasible solution is found much latter.

Heuristics motivational example (cont.)

Heuristics automatic

```
 499   385     2674.4232  1298                           2676.4137   511018
Elapsed real time =  86.54 sec. (tree size =  3.99 MB, solutions = 0)
    544   428    infeasible                              2676.4137   523630
*   604+  321                        -1.78665e+10       2672.4010   570690  100.00%
    604   322     2671.7797  1341    -1.78665e+10       2671.7797   577744  100.00%
    605   323     2671.5395  1410    -1.78665e+10       2671.7797   582540  100.00%
    608   324     2665.7742  1321    -1.78665e+10       2671.5025   589020  100.00%
    620   331     2670.9349  1440    -1.78665e+10       2671.2627   604374  100.00%
                                                         Cuts: 50
    640   339     2655.9561  1075    -1.78665e+10       2671.2627   653538  100.00%
                                                         Cuts: 25
*   658+  247                        -1.46007e+10       2671.2627   662376  100.00%
```

Heuristics turned off

```
9656  8295     2668.7217  1328                           2669.6422  3285646
Elapsed real time = 401.82 sec. (tree size = 824.86 MB, solutions = 0)
Nodefile size = 673.26 MB (610.47 MB after compression)
   9936  8567     2657.0882  1101                        2669.6422  3350837
  10472  9069    infeasible                              2669.6422  3475525
  10856  9420    infeasible                              2669.6422  3551932
* 11283  6532      integral     0    -9.18449e+10       2669.6422  3649047  100.00%
```