Planning & Scheduling by Logic-based Benders Decomposition: A Computational Analysis

André Ciré
University of Toronto

Elvin Çoban
Özyeğin University

John Hooker
Carnegie Mellon University

COPLAS 2015

Objective

- Identify factors that **explain the efficiency** of Logic-based Benders decomposition (LBBD) for planning and scheduling.
  - LBBD has brought **orders-of-magnitude improvement** over state of the art in several problem domains.
  - Factors that explain this success **have not been studied systematically**.

Test Problem

- As a test case, we use a simple **resource assignment and scheduling** problem.
  - Assign tasks to resources.
  - Schedule tasks assigned to each resource.
    - Tasks may run concurrently, subject to limit on total rate of resource consumption (cumulative scheduling).

What Is Logic-Based Benders?

- **Logic-based Benders decomposition** is a generalization of classical Benders decomposition.
  - Subproblem is an **arbitrary optimization problem**.
    - need not have linear or inequality model.
    - JH (1995), JH and Yan (1995), JH and Ottosson (2003).
  - Solves a problem of the form

$$\begin{aligned} &\min f(x,y) \\ &(x,y) \in S \\ &x \in D \end{aligned}$$

Logic-Based Benders

- Decompose problem into master and subproblem.
  - Subproblem is obtained by fixing x to solution value in master problem.

Master problem

$$\min z$$
$$z \geq g_k(x) \quad \text{(Benders cuts)}$$
$$x \in D$$

Minimize cost z subject to bounds given by Benders cuts, obtained from values of x attempted in previous iterations k.

→ Trial value $\bar{x}$ that solves master

← Benders cut $z \geq g_k(x)$

Subproblem

$$\min f(\bar{x}, y)$$
$$(\bar{x}, y) \in S$$

Obtain proof of optimality (solution of **inference dual**). Use same proof to deduce cost bounds for other assignments, yielding Benders cut.

Logic-Based Benders

- Iterate until master problem value equals best subproblem value so far.
  - This yields optimal solution.

Master problem

$\min z$

$z \geq g_k(x)$ (Benders cuts)

$x \in D$

Minimize cost z subject to bounds given by Benders cuts, obtained from values of x attempted in previous iterations k.

→ Trial value $\bar{x}$ that solves master

← Benders cut $z \geq g_k(x)$

Subproblem

$\min f(\bar{x}, y)$

$(\bar{x}, y) \in S$

Obtain proof of optimality (solution of **inference dual**). Use same proof to deduce cost bounds for other assignments, yielding Benders cut.

Logic-Based Benders

- Fundamental concept: **inference duality**

Primal problem:
optimization

$$\min f(x)$$
$$x \in S$$

Find **best** feasible solution by searching over **values of *x***.

Dual problem:
Inference

$$\max v$$
$$x \in S \overset{P}{\Longrightarrow} f(x) \geq v$$
$$P \in \mathcal{P}$$

Find a proof of optimal value *v** by searching over **proofs *P***.

Logic-Based Benders

- Popular optimization duals are **special cases** of the inference dual.
  - Result from different choices of **inference method**.
  - For example....
    - Linear programming dual (gives **classical Benders cuts**)
    - Lagrangean dual
    - Surrogate dual
    - Subadditive dual

Logic-Based Benders

- Substantial speedup for many applications.
  - Several orders of magnitude relative to state of the art.

Logic-Based Benders

- Substantial speedup for many applications.
  - Several orders of magnitude relative to state of the art.
- Some applications:
  - Circuit verification
  - Chemical batch processing (BASF, etc.)
  - Steel production scheduling
  - Auto assembly line management (Peugeot-Citroën)
  - Automated guided vehicles in flexible manufacturing
  - Allocation and scheduling of multicore processors (IBM, Toshiba, Sony)
  - Facility location-allocation
  - Stochastic facility location and fleet management
  - Capacity and distance-constrained plant location

Logic-Based Benders

- Some applications…
  - Transportation network design
  - Traffic diversion around blocked routes
  - Worker assignment in a queuing environment
  - Single- and multiple-machine allocation and scheduling
  - Permutation flow shop scheduling with time lags
  - Resource-constrained scheduling
  - Wireless local area network design
  - Service restoration in a network
  - Optimal control of dynamical systems
  - Sports scheduling

Application to Planning & Scheduling

- Assign tasks in master, schedule in subproblem.
  - Combine **mixed integer programming** and **constraint programming**

Master problem

Assign tasks to resources to minimize cost.

Solve by **mixed integer programming**.

→ Trial assignment $\bar{x}$ →

← Benders cut $z \geq g_k(x)$ ←

Subproblem

Schedule jobs on each machine, subject to time windows.

Constraint programming obtains proof of optimality (dual solution).

Use **same proof** to deduce cost for some other assignments, yielding Benders cut.

Application to Planning & Scheduling

- Objective function
  - Cost is based on **task assignment only**.

$$\text{cost} = \sum_{ij} c_{ij} x_{ij}, \quad x_{ij} = 1 \text{ if task } j \text{ assigned to resource } i$$

  - So cost appears only in the **master problem**.
  - Scheduling subproblem is a **feasibility problem**.

Application to Planning & Scheduling

- Objective function
  - Cost is based on **task assignment only**.

    $$\text{cost} = \sum_{ij} c_{ij} x_{ij}, \quad x_{ij} = 1 \text{ if task } j \text{ assigned to resource } i$$

    - So cost appears only in the **master problem**.
    - Scheduling subproblem is a **feasibility problem**.

- Benders cuts
  - They have the form $\sum_{j \in J_i} (1 - x_{ij}) \geq 1, \text{ all } i$
    - where J_i is a set of tasks that create infeasibility when assigned to resource i.

Application to Planning & Scheduling

- Resulting Benders decomposition:

Master problem

$$\min\ z$$

$$z = \sum_{ij} c_{ij} x_{ij}$$

Benders cuts

Trial assignment $\bar{x}$ →

← Benders cuts

$$\sum_{j \in J_i} (1 - x_{ij}) \geq 1,$$

for infeasible resources *i*

Subproblem

Schedule jobs on each resource.

Constraint programming may obtain proof of infeasibility on some resources (dual solution).

Use **same proof** to deduce infeasibility for some other assignments, yielding Benders cut.

Application to Planning & Scheduling

- Problem: We don't **have access** to infeasibility proof in CP solver.
  - So begin with simple **nogood cut** $\sum_{j \in J_i} (1 - x_{ij}) \geq 1, \;\; \text{all } i$

    where J_i contains all tasks assigned resource i.
  - Then **strengthen cut** by heuristically removing tasks from J_i until schedule on resource i becomes feasible.

Problem Instances

- Used in several previous studies.
  - Randomly generated near phase transition.
  - Schedule *n* tasks on *m* resources.
  - 5 instances for each (*m*,*n*) pair.

Problem Instances

- “c” instances
  - **10-32** tasks on **2-4** resources.
  - Designed to be **hard** for LBBD.
    - Resources differ by factor of *m* in processing speed.
  - Results in many tasks assigned to fastest resource, creating a computational bottleneck.

Problem Instances

- "c" instances
  - **10-32** tasks on **2-4** resources.
  - Designed to be **hard** for LBBD.
    - Resources differ by factor of *m* in processing speed.
  - Results in many tasks assigned to fastest resource, creating a computational bottleneck.
- "e" instances
  - **10-50** tasks on **2-10** resources.
  - Perhaps more realistic.
    - Resources differ by factor of *2* in processing speed.

Experimental Design

- Solve with LBBD
  - Using "**strong**" Benders cuts only
    - Strengthened nogood cuts.
  - Using "**weak**" cuts with subproblem **relaxation** in master.
    - Simple nogood cuts.
    - Relaxation: limit total "energy consumption" to energy available within span of time windows.
    - Energy = duration x resource consumption rate.
  - Using "**strong**" cuts with **relaxation**.
- Solve with mixed integer programming (MIP)
  - Using state-of-the-art commercial solver.
    - And best known MIP model.

Experimental Design

- Solvers
  - CPLEX 12.4.01 for master problem.
  - IBM CP Optimizer 12.4.01 for subproblem.
    - Extended filtering, DFS search
    - Default variable and value selection.
  - CPLEX 12.4.01 for pure MIP solution.
  - No comparison with pure CP solver
    - Previous experience shows it to be much slower than MIP.

“c” instances, 2 resources

Size		MIP (CPLEX)		LBBD: strong cuts only		LBBD: relax + weak cuts		LBBD: relax + strong cuts	
m	n	Solved	Sec	Solved	Sec	Solved	Sec	Solved	Sec
2	10	5	0.1	5	0.2	5	0.1	5	0.1
	12	5	0.2	5	0.2	5	0.1	5	0.0
	14	5	0.1	5	0.4	5	0.1	5	0.0
	16	5	28	5	2.0	5	0.2	5	0.3
	18	5	388	5	19	5	0.5	5	0.7
	20	4	1899	5	120	5	2.0	5	8.0
	22	3	3844+	4	1852+	5	617	5	955
	24	2	4346+	1	6341+	4	1495+	4	1936+
	26	1	6362+	0	-	5	327	4	1642+
	28	2	4384+	0	-	5	1004	5	1133
	30	0	-	0	-	2	5391+	2	5761+
	32	1	5813+	0	-	2	4325+	2	4325+

+ Computation terminated after 7200 sec for instances not solved to optimality.

“c” instances, 3 resources

Size		MIP (CPLEX)		LBBD: strong cuts only		LBBD: relax + weak cuts		LBBD: relax + strong cuts	
m	n	Solved	Sec	Solved	Sec	Solved	Sec	Solved	Sec
3	10	5	0.0	5	0.2	5	0.1	5	0.1
	12	5	0.1	5	0.4	5	0.5	5	0.1
	14	5	0.3	5	1.2	5	0.3	5	0.2
	16	5	13	5	5.6	5	2.7	5	0.8
	18	5	548	5	22	5	7.8	5	1.4
	20	4	1712+	5	30	5	1.2	5	0.5
	22	3	3674+	5	59	5	7.5	5	2.6
	24	2	4411+	4	1739+	5	15	5	5.7
	26	0	-	4	3510+	5	191	5	98
	28	2	5238+	2	6645+	5	270	5	209
	30	0	-	0	-	4	2354+	4	1856+
	32	0	-	0	-	2	4667+	2	4751+

+ Computation terminated after 7200 sec for instances not solved to optimality.

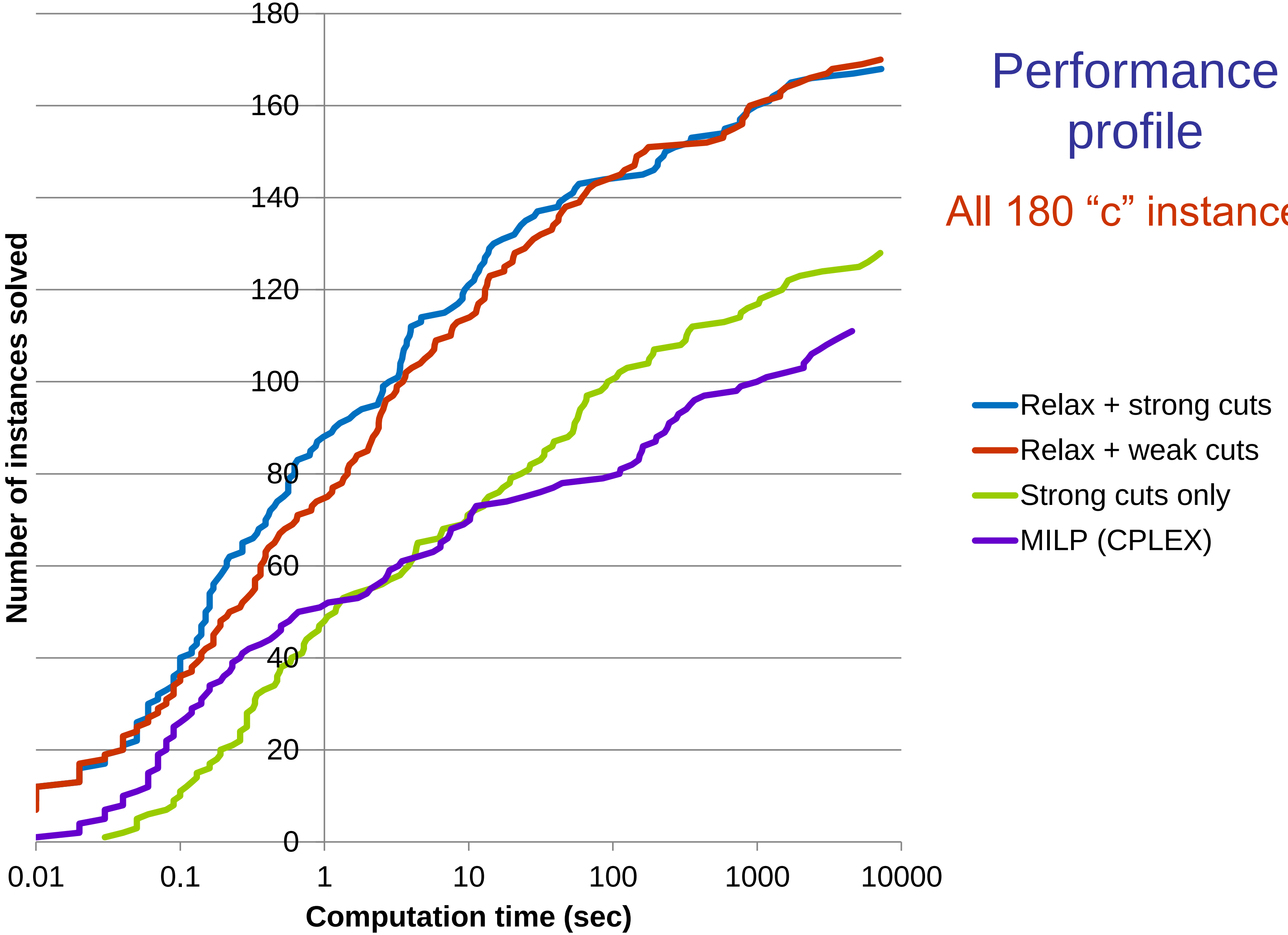
“c” instances, 4 resources

Size		MIP (CPLEX)		LBBD: strong cuts only		LBBD: relax + weak cuts		LBBD: relax + strong cuts	
m	n	Solved	Sec	Solved	Sec	Solved	Sec	Solved	Sec
4	10	5	0.0	5	0.1	5	0.0	5	0.0
	12	5	0.1	5	0.2	5	0.1	5	0.1
	14	5	0.3	5	0.6	5	1.0	5	0.3
	16	5	1.0	5	0.6	5	0.4	5	0.1
	18	5	36	5	4.0	5	1.7	5	0.4
	20	5	523	5	11	5	1.1	5	0.3
	22	5	811	5	75	5	8.2	5	1.1
	24	1	6292+	5	122	5	23	5	9.1
	26	0	-	3	3369+	5	19	5	7.4
	28	1	5762+	3	4623+	5	36	5	11
	30	0	-	2	4841+	5	430	5	61
	32	0	-	0	-	5	680	5	478

+ Computation terminated after 7200 sec for instances not solved to optimality.

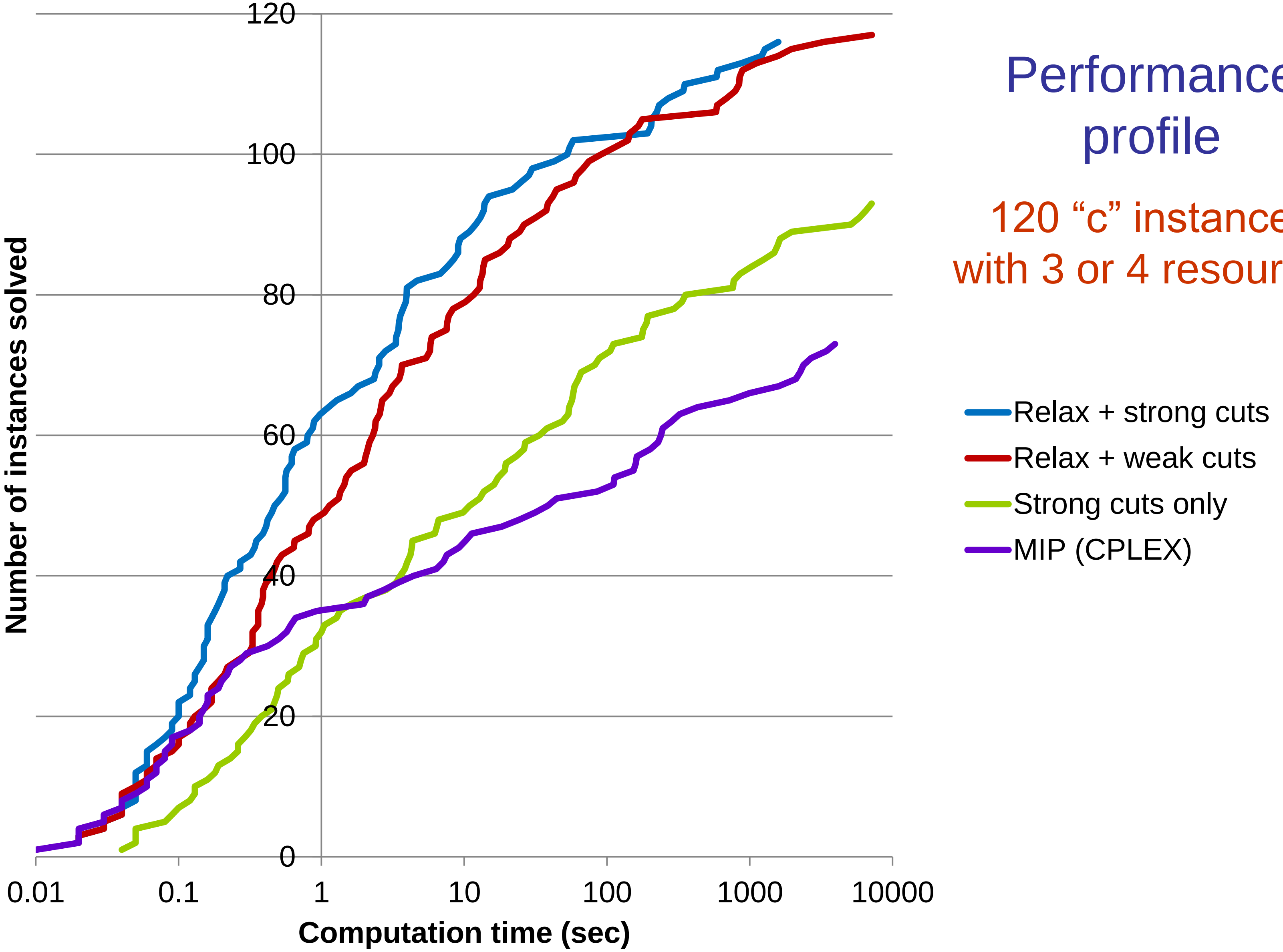
Performance profile

All 180 "c" instances



Performance profile

120 “c” instances with 3 or 4 resources



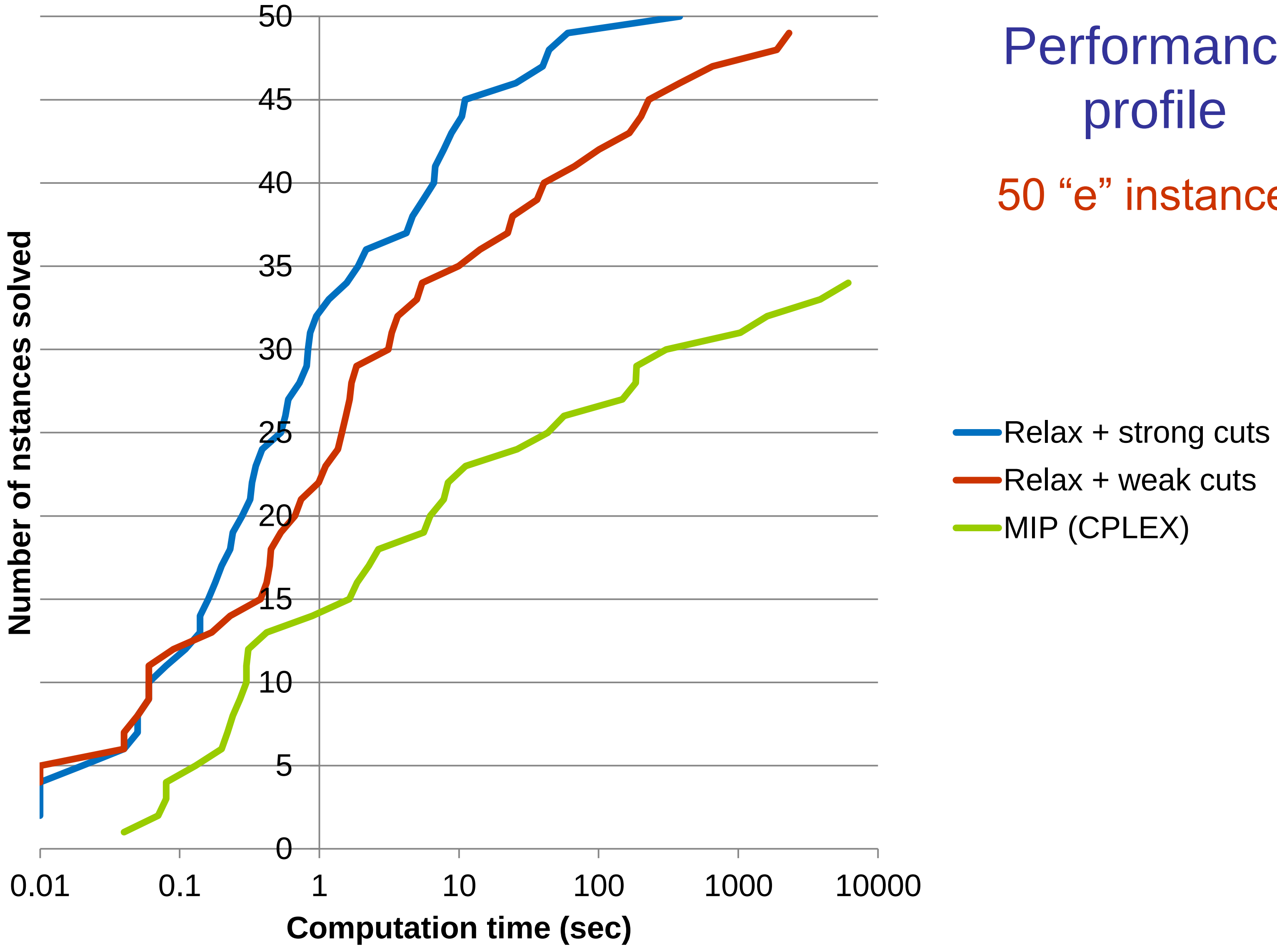
"e" instances

Size		MIP (CPLEX)		LBBD: relax + weak cuts		LBBD: relax + strong cuts	
m	n	Solved	Sec	Solved	Sec	Solved	Sec
2	10	5	0.1	5	0.1	5	0.1
2	12	5	0.3	5	0.3	5	0.1
3	15	5	0.9	5	0.4	5	0.2
4	20	5	46	5	14	5	1.9
5	25	5	73	5	1.0	5	0.7
6	30	5	543	5	1.3	5	0.4
7	35	2	5122+	5	36	5	2.7
8	40	1	7246+	4	1527+	5	80
9	45	0	-	5	1050	5	35
10	50	1	6983+	5	45	5	5.4

+ Computation terminated after 7200 sec for instances not solved to optimality.

Performance profile

50 “e” instances



Observations

- LBBD is orders of magnitude faster than MIP.
  - Less dramatic for “c” instances with 2 resources.
    - Almost all complexity is in the subproblem.
- Relaxation is most important success factor.
- Strength of cut is important for larger instances.
  - Especially for “e” instances.

Further Analysis

- Number of Benders iterations
- Breakdown of computation time
  - Master problem vs. subproblem

Relaxation reduces number of iterations and therefore difficulty of master problem.

"c" instances, 2 resources

		Strong cuts only			Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec		sec	sec
2	10	18	0.1	0.1	9.8	0.1	0.0	4.8	0.0	0.0
	12	13	0.1	0.1	5.0	0.0	0.0	3.4	0.0	0.0
	14	19	0.1	0.3	1.8	0.0	0.0	1.8	0.0	0.0
	16	41	0.5	1.5	2.0	0.0	0.2	2.0	0.0	0.3
	18	149	5.7	14	2.4	0.0	0.5	2.4	0.0	0.7
	20	107	3.5	117	3.6	0.0	2.0	2.8	0.0	8.0
	22	340+	70+	1782+	4.6	0.0	617	4.4	0.0	955
	24	327+	67+	6263+	2.0+	0.0+	1495+	1.8+	0.0+	1936+
	26	-	-	-	1.8	0.0	327	1.6+	0.0+	1642+
	28	-	-	-	2.0	0.0	1004	1.8	0.0	1133
	30	-	-	-	4.2+	0.0+	5391+	1.0+	1452+	4309+
	32	-	-	-	1.2+	0.0+	4325+	1.0+	0.0+	4325+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Relaxation reduces number of iterations and therefore difficulty of master problem.

"c" instances, 3 resources

m	n	Strong cuts only Iters	Master sec	Subpr sec	Relax + weak cuts Iters	Master sec	Subpr sec	Relax + strong cuts Iters	Master sec	Subpr sec
3	10	13	0.0	0.1	9.8	0.1	0.0	4.4	0.0	0.0
	12	23	0.2	0.2	14	0.4	0.0	6.4	0.1	0.1
	14	42	0.7	0.5	13	0.2	0.1	6.8	0.1	0.1
	16	86	4.0	1.5	40	2.5	0.2	17	0.5	0.3
	18	183	19	3.0	61	7.3	0.5	23	1.0	0.5
	20	226	23	6.4	21	0.8	0.4	8.2	0.1	0.4
	22	340	49	10	49	2.9	4.6	16	0.4	2.3
	24	1222+	1689+	50+	55	12	3.5	22	1.6	4.1
	26	1854+	2723+	786+	130	33	158	22	0.6	97
	28	2113+	3283+	3363+	15	0.2	270	8.0	0.1	209
	30	-	-	-	80+	9.2+	2344+	21+	1.1+	1855+
	32	-	-	-	143+	64+	4602+	23+	1.7+	4750+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Relaxation reduces number of iterations and therefore difficulty of master problem.

“c” instances, 4 resources

m	n	Strong cuts only Iters	Strong cuts only Master sec	Strong cuts only Subpr sec	Relax + weak cuts Iters	Relax + weak cuts Master sec	Relax + weak cuts Subpr sec	Relax + strong cuts Iters	Relax + strong cuts Master sec	Relax + strong cuts Subpr sec
4	10	6.8	0.0	0.1	4.6	0.0	0.0	3.0	0.0	0.0
	12	12	0.1	0.1	6.2	0.1	0.0	4.4	0.0	0.0
	14	26	0.3	0.3	22	0.9	0.1	9.0	0.2	0.1
	16	27	0.2	0.3	12	0.3	0.1	5.6	0.1	0.1
	18	74	3.0	1.0	32	1.5	0.1	15	0.3	0.2
	20	130	9.0	2.3	26	1.0	0.2	11	0.1	0.2
	22	334	69	6.6	51	7.6	0.6	15	0.7	0.5
	24	407	104	18	96	20	3.1	37	3.4	5.6
	26	1351+	3315+	54+	83	11	7.5	32	2.0	5.4
	28	2042+	4091+	532+	27	1.7	34	12	0.5	11
	30	1408+	4665+	175+	117	395	35	41	41	20
	32	-	-	-	60	6.3	673	14	0.4	478

+ Computation terminated after 7200 sec for instances not solved to optimality.

Severe imbalance of master and subproblem time, resulting in poorer performance for LBBD.

"c" instances, 2 resources

		Strong cuts only			Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec		sec	sec
2	10	18	0.1	0.1	9.8	0.1	0.0	4.8	0.0	0.0
	12	13	0.1	0.1	5.0	0.0	0.0	3.4	0.0	0.0
	14	19	0.1	0.3	1.8	0.0	0.0	1.8	0.0	0.0
	16	41	0.5	1.5	2.0	0.0	0.2	2.0	0.0	0.3
	18	149	5.7	14	2.4	0.0	0.5	2.4	0.0	0.7
	20	107	3.5	117	3.6	0.0	2.0	2.8	0.0	8.0
	22	340+	70+	1782+	4.6	0.0	617	4.4	0.0	955
	24	327+	67+	6263+	2.0+	0.0+	1495+	1.8+	0.0+	1936+
	26	-	-	-	1.8	0.0	327	1.6+	0.0+	1642+
	28	-	-	-	2.0	0.0	1004	1.8	0.0	1133
	30	-	-	-	4.2+	0.0+	5391+	1.0+	1452+	4309+
	32	-	-	-	1.2+	0.0+	4325+	1.0+	0.0+	4325+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Subproblem blows up when more than 10 tasks per resource **on average**

"c" instances, 2 resources

m	n	Strong cuts only Iters	Strong cuts only Master sec	Strong cuts only Subpr sec	Relax + weak cuts Iters	Relax + weak cuts Master sec	Relax + weak cuts Subpr sec	Relax + strong cuts Iters	Relax + strong cuts Master sec	Relax + strong cuts Subpr sec
2	10	18	0.1	0.1	9.8	0.1	0.0	4.8	0.0	0.0
	12	13	0.1	0.1	5.0	0.0	0.0	3.4	0.0	0.0
	14	19	0.1	0.3	1.8	0.0	0.0	1.8	0.0	0.0
	16	41	0.5	1.5	2.0	0.0	0.2	2.0	0.0	0.3
	18	149	5.7	14	2.4	0.0	0.5	2.4	0.0	0.7
	20	107	3.5	117	3.6	0.0	2.0	2.8	0.0	8.0
	22	340+	70+	1782+	4.6	0.0	617	4.4	0.0	955
	24	327+	67+	6263+	2.0+	0.0+	1495+	1.8+	0.0+	1936+
	26	-	-	-	1.8	0.0	327	1.6+	0.0+	1642+
	28	-	-	-	2.0	0.0	1004	1.8	0.0	1133
	30	-	-	-	4.2+	0.0+	5391+	1.0+	1452+	4309+
	32	-	-	-	1.2+	0.0+	4325+	1.0+	0.0+	4325+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Subproblem blows up when more than 10 tasks per resource **on average**

"c" instances, 3 resources

		Strong cuts only			Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec		sec	sec
3	10	13	0.0	0.1	9.8	0.1	0.0	4.4	0.0	0.0
	12	23	0.2	0.2	14	0.4	0.0	6.4	0.1	0.1
	14	42	0.7	0.5	13	0.2	0.1	6.8	0.1	0.1
	16	86	4.0	1.5	40	2.5	0.2	17	0.5	0.3
	18	183	19	3.0	61	7.3	0.5	23	1.0	0.5
	20	226	23	6.4	21	0.8	0.4	8.2	0.1	0.4
	22	340	49	10	49	2.9	4.6	16	0.4	2.3
	24	1222+	1689+	50+	55	12	3.5	22	1.6	4.1
	26	1854+	2723+	786+	130	33	158	22	0.6	97
	28	2113+	3283+	3363+	15	0.2	270	8.0	0.1	209
	30	-	-	-	80+	9.2+	2344+	21+	1.1+	1855+
	32	-	-	-	143+	64+	4602+	23+	1.7+	4750+

+ Computation terminated after 7200 sec for instances not solved to optimality.

Balance between master and subproblem results in superior performance

"e" instances

m	n	Relax + weak cuts Iters	Relax + weak cuts Master sec	Relax + weak cuts Subpr sec	Relax + strong cuts Iters	Relax + strong cuts Master sec	Relax + strong cuts Subpr sec
2	10	9.4	0.1	0.0	5.2	0.0	0.0
2	12	13	0.3	0.0	4.4	0.0	0.0
3	15	14	0.4	0.0	5.6	0.1	0.1
4	20	55	14	0.0	16	1.7	0.3
5	25	19	0.4	0.0	8.6	0.1	0.6
5	30	26	1.1	0.0	8.8	0.2	0.2
7	35	76	34	0.0	19	2.0	0.7
8	40	107+	1525+	0.0+	31	78	2.1
9	45	132	1048	0.0	39	33	2.2
10	50	39	43	0.0	18	3.6	1.7

+ Computation terminated after 7200 sec for instances not solved to optimality.

Mild imbalance results in somewhat worse performance

"e" instances

		Relax + weak cuts			Relax + strong cuts		
		Iters	Master	Subpr	Iters	Master	Subpr
m	n		sec	sec		sec	sec
2	10	9.4	0.1	0.0	5.2	0.0	0.0
2	12	13	0.3	0.0	4.4	0.0	0.0
3	15	14	0.4	0.0	5.6	0.1	0.1
4	20	55	14	0.0	16	1.7	0.3
5	25	19	0.4	0.0	8.6	0.1	0.6
5	30	26	1.1	0.0	8.8	0.2	0.2
7	35	76	34	0.0	19	2.0	0.7
8	40	107+	1525+	0.0+	31	78	2.1
9	45	132	1048	0.0	39	33	2.2
10	50	39	43	0.0	18	3.6	1.7

+ Computation terminated after 7200 sec for instances not solved to optimality.

Conclusions

- Superiority to MIP, CP
  - LBBD remains **orders of magnitude** faster than MIP.
    - For this problem class.
    - Despite huge improvements in MIP.
    - Even greater advantage over CP.

Conclusions

- Superiority to MIP, CP
  - LBBD remains **orders of magnitude** faster than MIP.
    - For this problem class.
    - Despite huge improvements in MIP.
    - Even greater advantage over CP.
- Importance of master/subproblem balance
  - LBBD is most effective when master and subproblem share substantial portions of combinatorial complexity.
    - …and consume **roughly equal time**.

Conclusions

- Superiority to MIP, CP
  - LBBD remains **orders of magnitude** faster than MIP.
    - For this problem class.
    - Despite huge improvements in MIP.
    - Even greater advantage over CP.
- Importance of master/subproblem balance
  - LBBD is most effective when master and subproblem share substantial portions of combinatorial complexity.
    - …and consume **roughly equal time**.
- Failure in subproblem
  - LBBD most often fails when **subproblem blows up** due to assignment of too many tasks to a resource.

Conclusions

- Subproblem relaxation
  - **Most important success factor** for LBBD is inclusion of a subproblem relaxation in the master.

Conclusions

- Subproblem relaxation
  - **Most important success factor** for LBBD is inclusion of a subproblem relaxation in the master.
- Strong Benders cuts
  - Stronger Benders cuts can help significantly when master and subproblem are **properly balanced** in difficulty.

Suggested Solution Strategies

- Tighter subproblem relaxations
  - Design tighter subproblem relaxations for the master
    - …**using subproblem variables**, whose values are discarded after master is solved

Suggested Solution Strategies

- Tighter subproblem relaxations
  - Design tighter subproblem relaxations for the master
    - …**using subproblem variables**, whose values are discarded after master is solved
- Failure-directed search in subproblem
  - **Recently introduced** in ILOG CP optimizer.

Suggested Solution Strategies

- Tighter subproblem relaxations
  - Design tighter subproblem relaxations for the master
    - …**using subproblem variables**, whose values are discarded after master is solved
- Failure-directed search in subproblem
  - **Recently introduced** in ILOG CP optimizer.
- Subproblem decomposition
  - Solve subproblem with LBBD when it **grows too large**.

Suggested Solution Strategies

- Tighter subproblem relaxations
  - Design tighter subproblem relaxations for the master
    - …**using subproblem variables**, whose values are discarded after master is solved
- Failure-directed search in subproblem
  - **Recently introduced** in ILOG CP optimizer.
- Subproblem decomposition
  - Solve subproblem with LBBD when it **grows too large**.
- More dual information
  - Use subproblem solver that **reveals proof of optimality**, perhaps resulting in stronger Benders cuts.