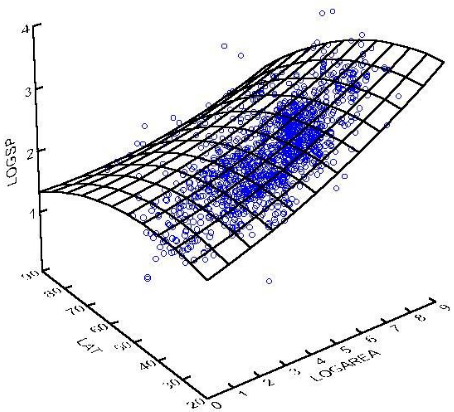
Learning-Based Methods for Large-Scale Optimization

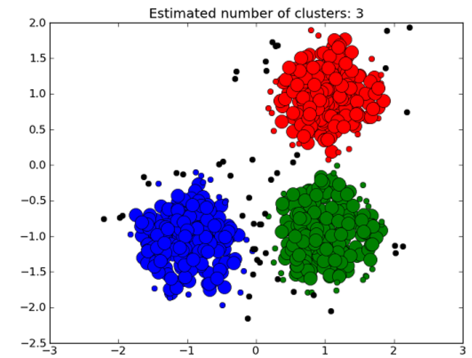
John Hooker
Carnegie Mellon University

Whitney Symposium
GE Global Research Center
October 2016

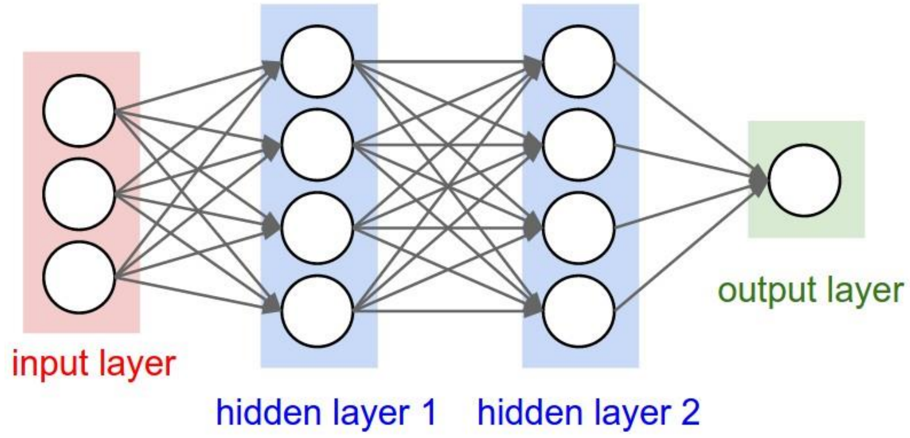
Learning and AI

- **Learning** lies at the core of artificial intelligence
  - Statistical methods
    - Regression, clustering, pattern recognition



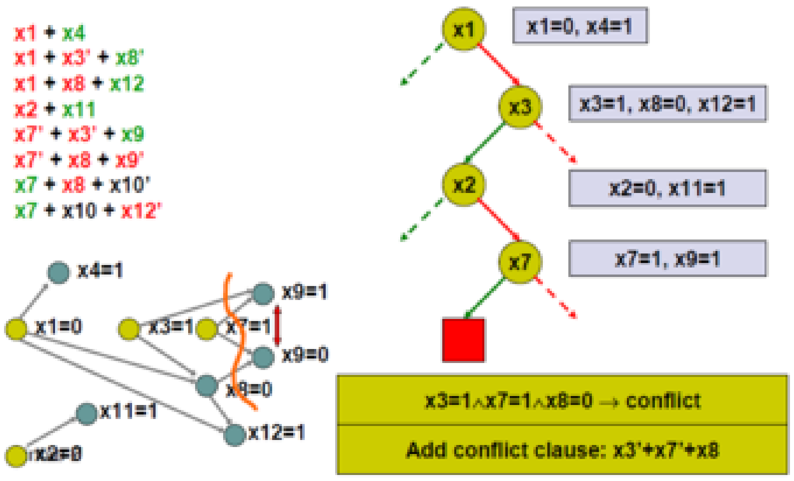


  - Neural networks
    - "Deep learning" in multilayer networks



Learning and AI

- **Another type of** learning has been developing in the optimization and constraint solving communities.
  - **Conflict-directed learning**



Learning and AI

- **Conflict-directed learning**
  - **Satisfiability** (SAT) solvers
    - Conflict-directed **clause learning**
    - Modern solvers can deal with huge problems
  - **Decomposition methods**
    - Conflict-directed constraint learning
    - Yields **logic-based Benders cuts**
    - Can break huge problems into smaller pieces
  - Common strategy: **learn from your mistakes**

Learning and AI

- **Conflict-directed learning**
  - **Satisfiability** (SAT) solvers
    - Conflict-directed **clause learning**
    - Modern solvers can deal with huge problems
  - **Decomposition methods**
    - Conflict-directed constraint learning
    - Yields **logic-based Benders cuts**
    - Can break huge problems into smaller pieces
  - Common strategy: **learn from your mistakes**

- This is all coming together
  - **Conflict clauses** = special case of **Benders cuts!**

Outline

- Learning in **satisfiability** (SAT) algorithms
  - Example: product configuration
  - Example: robot motion planning
  - Applications of SAT
  - SAT solvers
- Learning in **Benders decomposition**
  - Classical Benders decomposition
  - Logic-based Benders decomposition
  - Example: machine scheduling
  - Applications of logic-based Benders
  - Example: logical inference
  - Example: home health care
  - References

Outline

- Regarding the future of AI
  - Codifying ethics
  - Autonomous machines and ethics

Two SAT Applications

- Product configuration
  - Daimler-Benz
- Robot motion planning

Product Configuration

Options available for Mercedes-Benz's C class: (excerpt, total: 692)

231 garage door opener integrated into interior mirror
280 steering wheel in leather design (two-colored) with chrome clip
550 trailer appliance
581 comfort air-conditioning THERMOTRONIC
671 light metal wheels 4x, 7 spoke design
6
7
9

Restrictions for Mercedes-Benz's C class: (excerpt, total: 952)

AMG styling (772) cannot be combined with trailer appliance (550).

Comfort air-conditioning (581) requires high-capacity battery (673), except when combined with gasoline engines with 2.6 or 3.2 liter cylinder capacity.

Source: Carsten Sinz, Johannes Kepler University Linz

Product Configuration

Restrictions for Mercedes-Benz's C class: (excerpt, total: 952)

AMG styling (772) cannot be combined with trailer appliance (550).

Comfort air-conditioning (581) requires high-capacity battery (673), except when combined with gasoline engines with 2.6 or 3.2 liter cylinder capacity.

Configuration rule

$$\text{not } (772 \text{ and } 550)$$

Translation to logical clause(s)

$$\neg 772 \vee \neg 550$$

not

or

Product Configuration

Restrictions for Mercedes-Benz's C class: (excerpt, total: 952)

AMG styling (772) cannot be combined with trailer appliance (550).

Comfort air-conditioning (581) requires high-capacity battery (673), except when combined with gasoline engines with 2.6 or 3.2 liter cylinder capacity.

Configuration rule	Translation to logical clause(s)
not (772 and 550)	$\neg 772 \vee \neg 550$
if not(2.6L or 3.2L) then (581 implies 673)	$2.6\mathrm{L} \vee 3.2\mathrm{L} \vee \neg 581 \vee 673$

Product Configuration

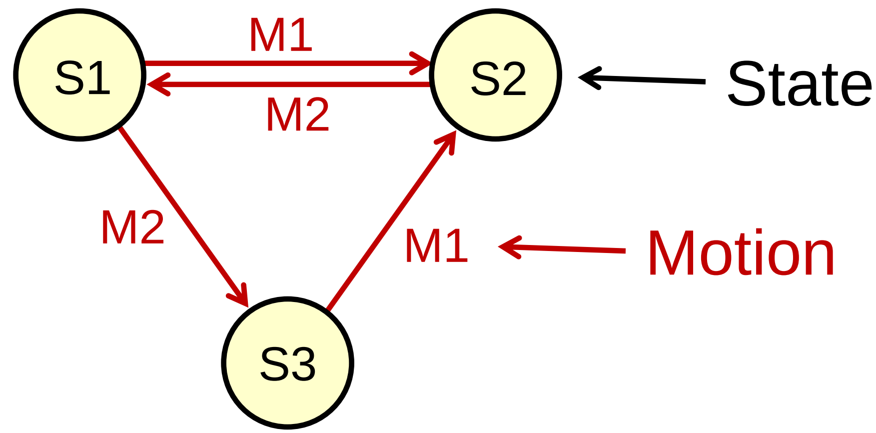
More realistic configuration rule:

((-L/(M111+M23+M001/M112+M28/M113)+-
(220/248/289/331/480/481/500/540/611/656/657+956/819/875+-(460/M113)/882/W10/Y94/Y95/X35/
X59/X62))+-R)+((-L/M113+-X62/M112+M28+-(772/774/X62)/M111+M23+M001+-(280+-
460/772/774/X62))+-R)+((-L/M112+M28+222+223+231+
254+292+423+(460/249+461+551+810)+(524+668+634+636/820)+543+581+679+(955+265+657
+(140A/200A)/956+570+(201A/208A))+809/M112+M28+221+222+231+254+292+(349/460)+423+
(460/249+461+551+810)+(524+668+634+636/820)+543+581+679+955+265+657+(140A/200A)+
800/M112+M28+221+222+231+254+292+(349/460)+423+(460/249+461+551+810)+(524+668+6
34+636/820)+543+581+679+956+570+(201A/208A)+800/M113+231+249+254+265+441+(460/46
1)+(551/460)+(810/460)+(524+668+634+636/820)+543+580A+809/M113+231+249+254+265+(34
9/460)+441+(460/461)+(551/460)+(810/460)+(524+668+634+636/820)+543+580A+800/M111+M2
3+M001+221+231+249+254+292+423+(524+634...
X34/X51/X52/X54/X55/X57/X58/X60/X61/X63/X64))

Combine rules and customer choices to form a clause set.
Solve a SAT problem to check if configuration is feasible.

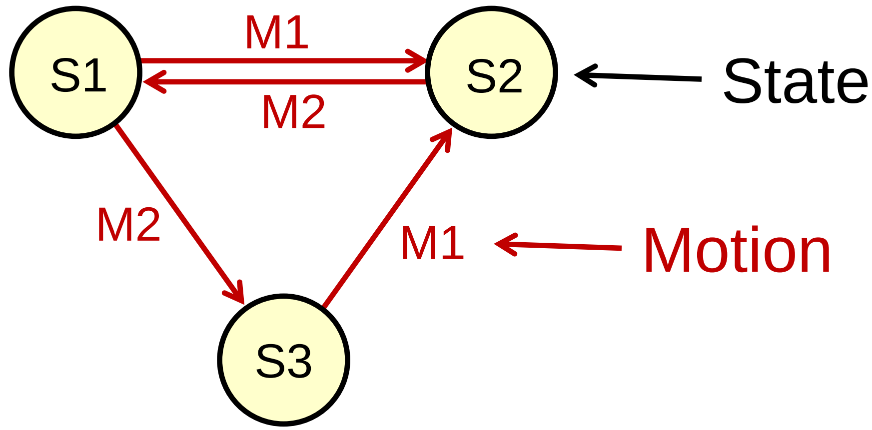
Robot Motion Planning

State transition graph:



Robot Motion Planning

State transition graph:



State transition rules

$$\text{if } \mathrm{M1}_t \text{ then } (\mathrm{S1}_t \text{ or } \boxed{\mathrm{S3}_t})$$

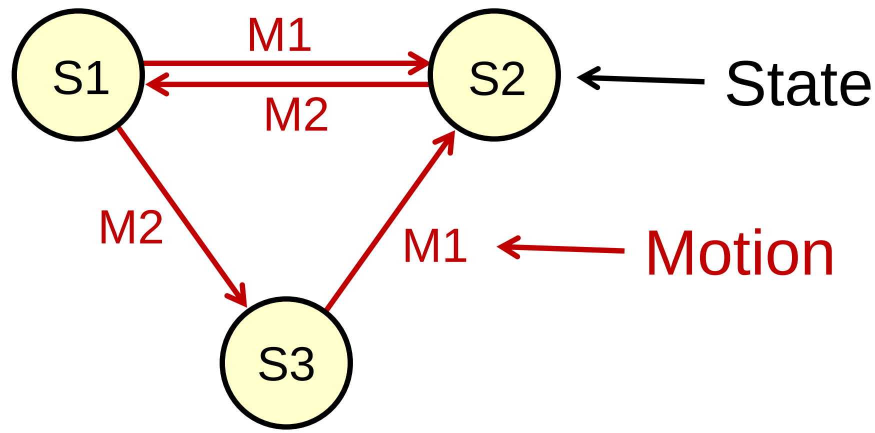
Robot in state S3 at time *t*

$$\text{if } \mathrm{M2}_t \text{ then } (\mathrm{S1}_t \text{ or } \mathrm{S2}_t)$$

$$\text{if } (\mathrm{S1}_t \text{ and } \mathrm{M1}_t) \text{ then } \mathrm{S2}_{t+1}$$
$$\text{if } (\mathrm{S1}_t \text{ and } \mathrm{M2}_t) \text{ then } \mathrm{S3}_{t+1}$$
$$\text{if } (\mathrm{S2}_t \text{ and } \mathrm{M2}_t) \text{ then } \mathrm{S1}_{t+1}$$
$$\text{if } (\mathrm{S3}_t \text{ and } \mathrm{M1}_t) \text{ then } \mathrm{S2}_{t+1}$$
$$\text{not}(\mathrm{M1}_t \text{ and } \mathrm{M2}_t)$$

Robot Motion Planning

State transition graph:



State transition rules

$\text{if } \mathrm{M1}_t \text{ then } (\mathrm{S1}_t \text{ or } \mathrm{S3}_t)$

$\text{if } \mathrm{M2}_t \text{ then } (\mathrm{S1}_t \text{ or } \mathrm{S2}_t)$

$\text{if } (\mathrm{S1}_t \text{ and } \mathrm{M1}_t) \text{ then } \mathrm{S2}_{t+1}$

$\text{if } (\mathrm{S1}_t \text{ and } \mathrm{M2}_t) \text{ then } \mathrm{S3}_{t+1}$

$\text{if } (\mathrm{S2}_t \text{ and } \mathrm{M2}_t) \text{ then } \mathrm{S1}_{t+1}$

$\text{if } (\mathrm{S3}_t \text{ and } \mathrm{M1}_t) \text{ then } \mathrm{S2}_{t+1}$

$\text{not}(\mathrm{M1}_t \text{ and } \mathrm{M2}_t)$

SAT clauses

$\neg \mathrm{M1}_t \vee \mathrm{S1}_t \vee \mathrm{S3}_t$

$\neg \mathrm{M2}_t \vee \mathrm{S1}_t \vee \mathrm{S2}_t$

$\neg \mathrm{S1}_t \vee \neg \mathrm{M1}_t \vee \mathrm{S2}_{t+1}$

$\neg \mathrm{S1}_t \vee \neg \mathrm{M2}_t \vee \mathrm{S3}_{t+1}$

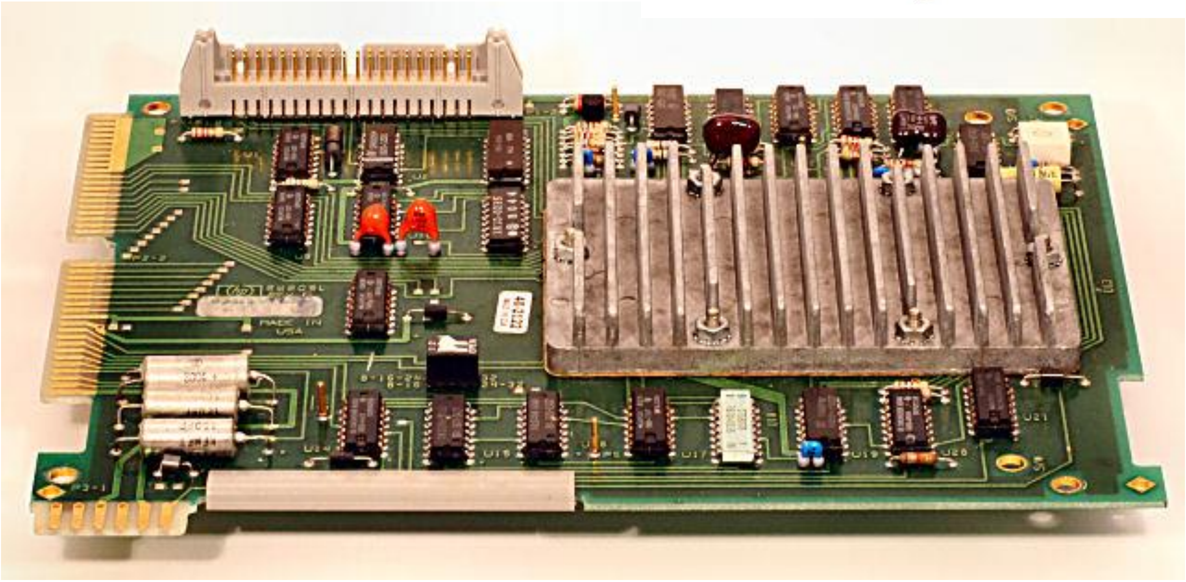
$\neg \mathrm{S2}_t \vee \neg \mathrm{M2}_t \vee \mathrm{S1}_{t+1}$

$\neg \mathrm{S3}_t \vee \neg \mathrm{M1}_t \vee \mathrm{S2}_{t+1}$

$\neg \mathrm{M1}_t \vee \neg \mathrm{M2}_t$

SAT Applications

- Model checking
  - Microprocessor verification
  - Software debugging
  - Device driver verification
  - Expert system verification



SAT Applications

- AI Planning
  - Robot motion planning
  - Autonomous vehicle control
  - Spacecraft mission planning
  - Equipment maintenance
  - Evacuation planning

SAT Applications

- Combinatorial design
  - Design of experiments
  - Cryptography
  - Drug design and testing
  - Crop rotation schedules

SAT Applications

- Product configuration
- Other applications
  - Test pattern generation
  - Traffic network scheduling
  - Optimal control
  - Multi-agent systems
  - Electronic auctions
  - Sports scheduling
  - Puzzle solving (e.g. sudoku)

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

SAT Solver Improvement

- Early 1990s:
  - 100 variables, 200 clauses
- Today:
  - > 1 million variables, > 5 million clauses
  - Mainly algorithmic improvements, not faster computers
- How is this possible?
  - Mainly due to **conflict-directed clause learning**
  - That is, generation of **conflict clauses**

Conflict Clauses

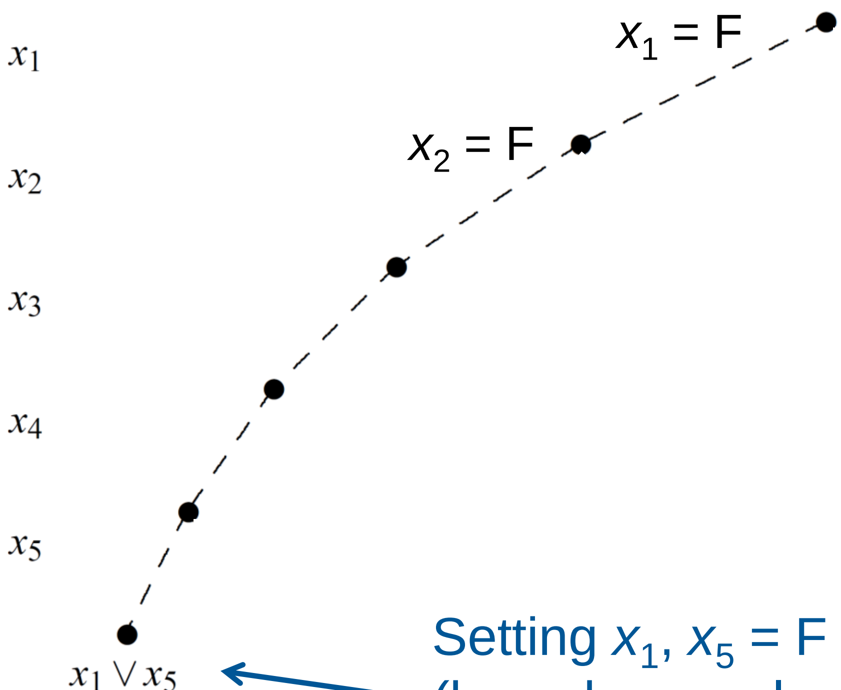
- We wish to solve the SAT instance

$$
\begin{array}{llllll}
x_1 & & & \vee x_4 & \vee x_5 & \\
x_1 & & & \vee x_4 & \vee \bar{x}_5 & \\
x_1 & & & & \vee x_5 & \vee x_6 \\
x_1 & & & & \vee x_5 & \vee \bar{x}_6 \\
 & x_2 & & & \vee \bar{x}_5 & \vee x_6 \\
 & x_2 & & & \vee \bar{x}_5 & \vee \bar{x}_6 \\
 & & x_3 & \vee \bar{x}_4 & \vee x_5 & \\
 & & x_3 & \vee \bar{x}_4 & \vee \bar{x}_5 &
\end{array}
$$

Find an assignment of True and False to variables x_j that satisfies all clauses

Conflict Clauses

Start branching on variables in depth-first fashion.
At each node of the branching tree, check if a clause is violated

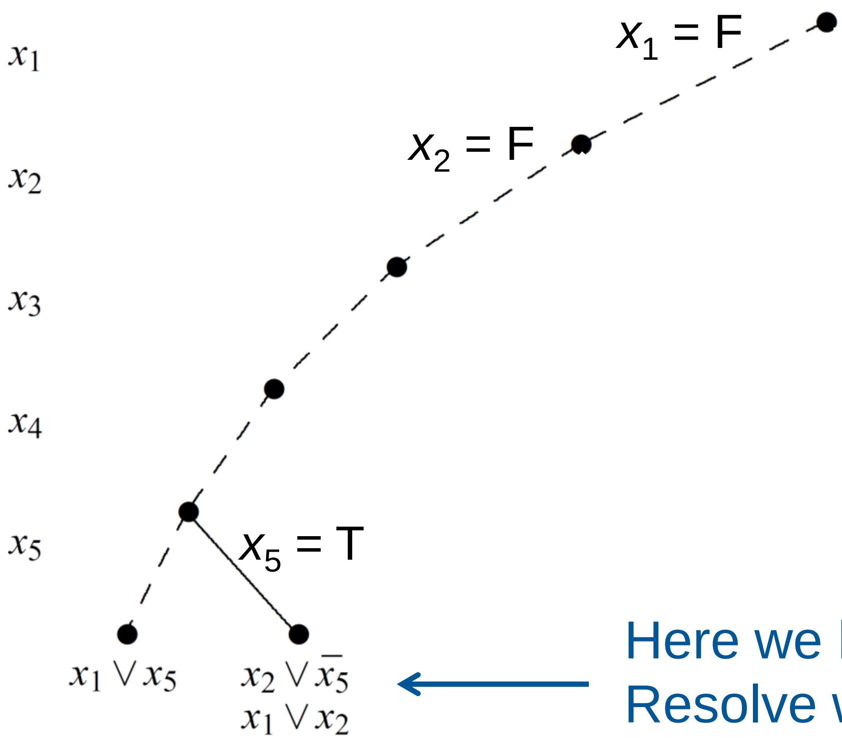


Violation

Setting x_1, x_5 = F causes the violation
(based on analysis of an implication graph)
So we add the **conflict clause** $x_1 \vee x_5$
to avoid this assignment in the future

Conflict Clauses

Start branching on variables in depth-first fashion.
At each node of the branching tree, check if a clause is violated

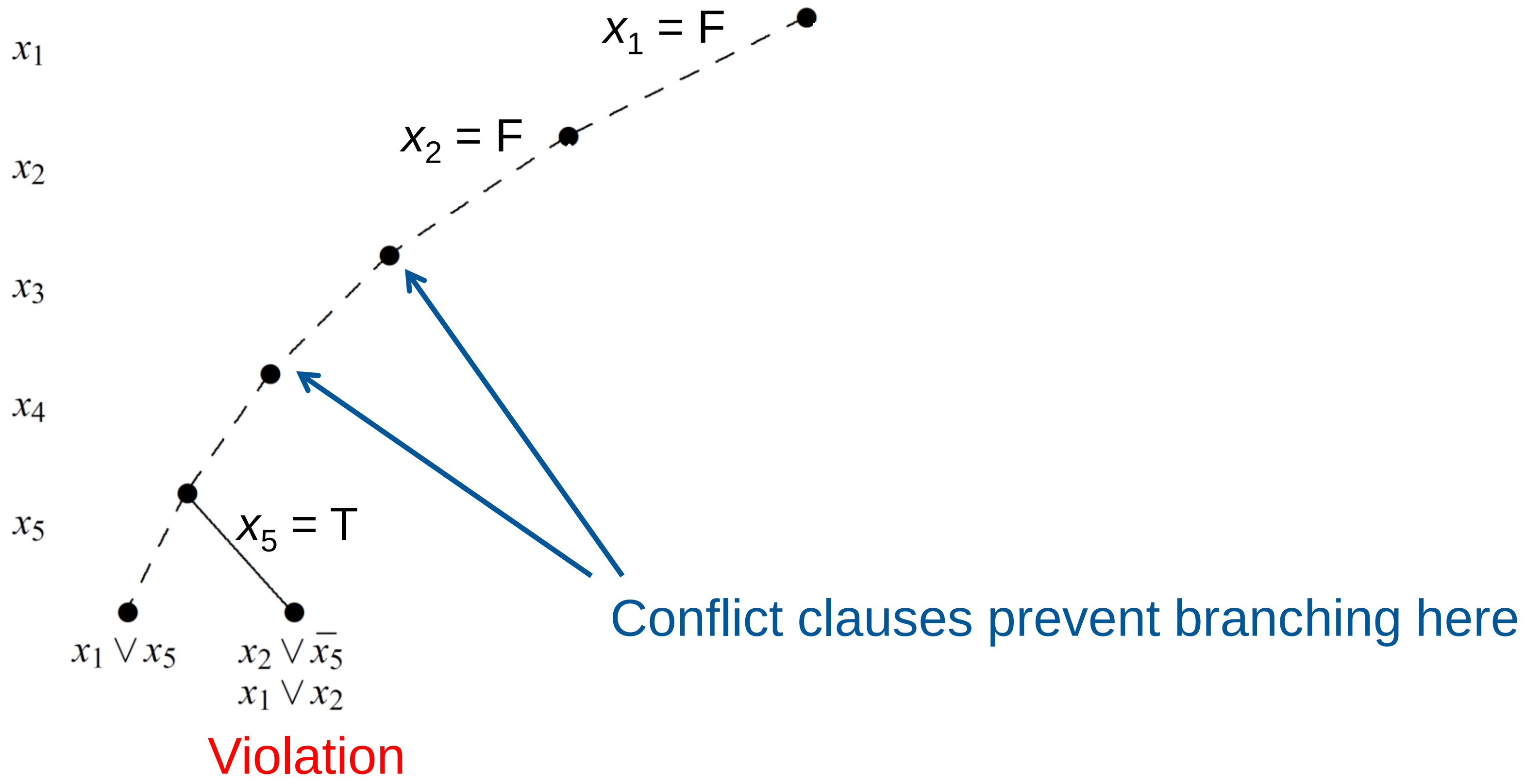


Violation

Here we have a **conflict clause** $x_1 \vee \neg x_5$
Resolve with $x_1 \vee x_2$ to get $x_1 \vee x_2$

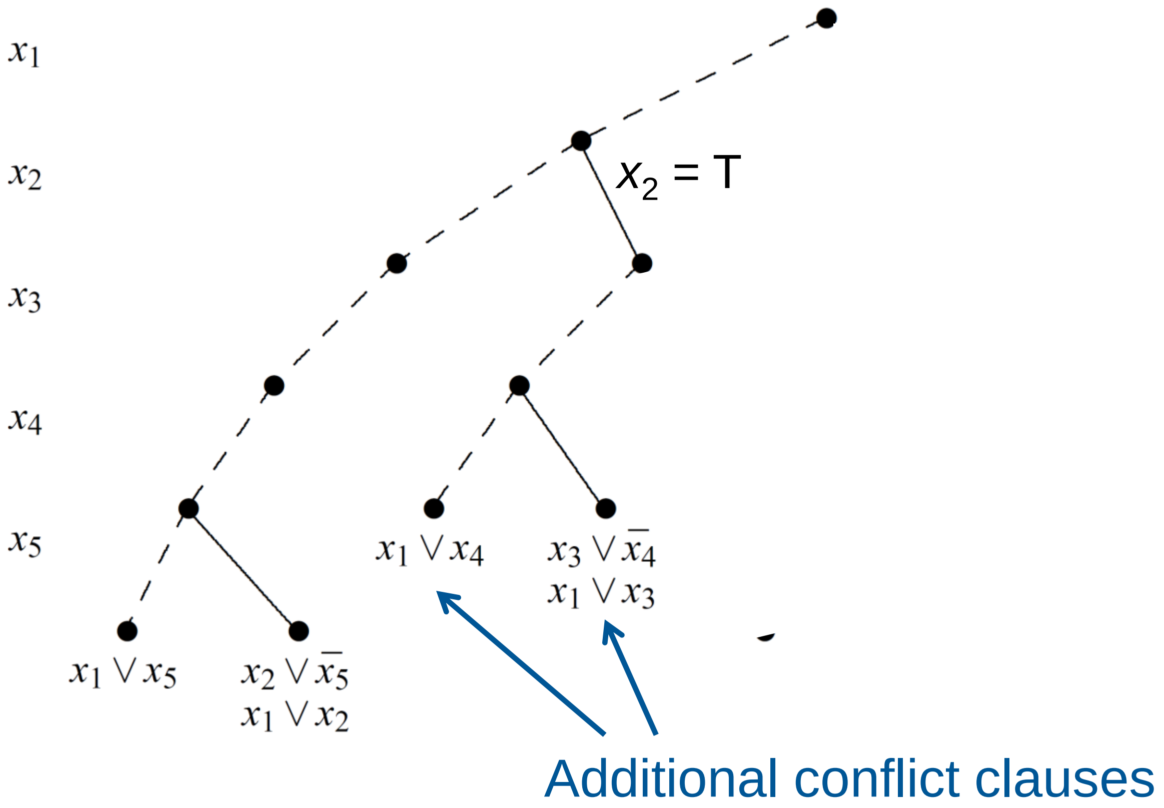
Conflict Clauses

Start branching on variables in depth-first fashion.
At each node of the branching tree, check if a clause is violated



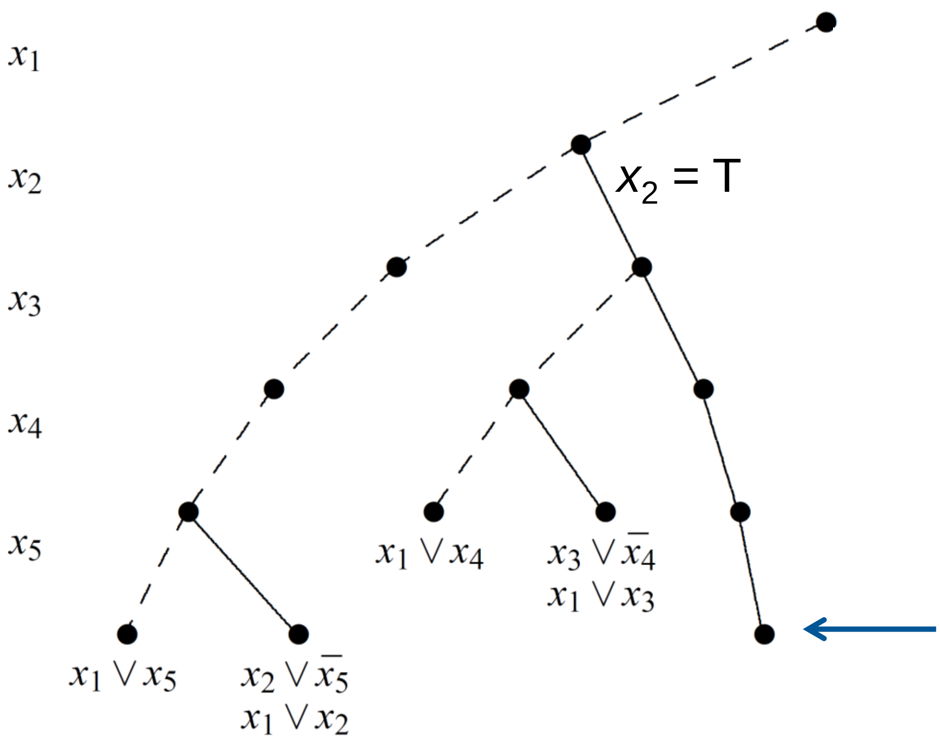
Conflict Clauses

Start branching on variables in depth-first fashion.
At each node of the branching tree, check if a clause is violated



Conflict Clauses

Start branching on variables in depth-first fashion.
At each node of the branching tree, check if a clause is violated



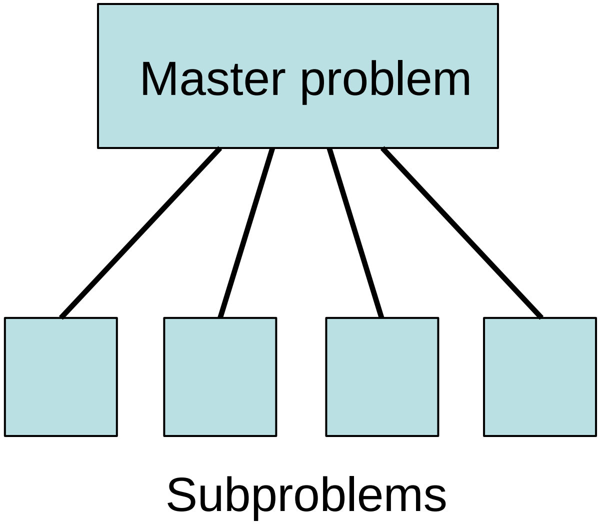
Solution found here
$x_1 = F$
$x_2 = x_3 = x_4 = x_5 = x_6 = T$

Decomposition

- **Decomposition** breaks a large problem into subproblems that can be solved separately.
  - But with some kind of **communication** among the subproblems.
  - Decomposition is an **essential strategy** for solving today's ever larger and more interconnected models.

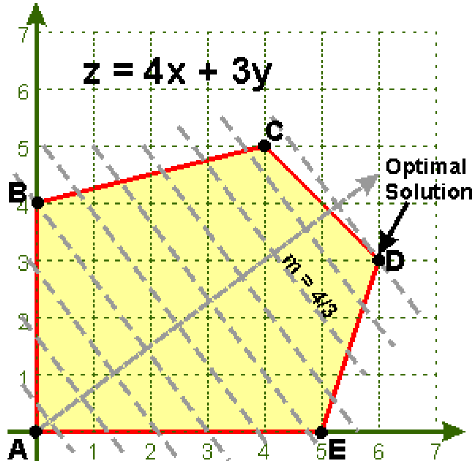
Benders Decomposition

- **Benders decomposition** is a classical strategy that does not sacrifice overall optimality.
  - Separates the problem into a **master problem** and multiple **subproblems**.
    - Variables are partitioned between master and subproblems.
    - Exploits the fact that the problem may **radically simplify** when the master problem variables are fixed to a set of values.



Benders Decomposition

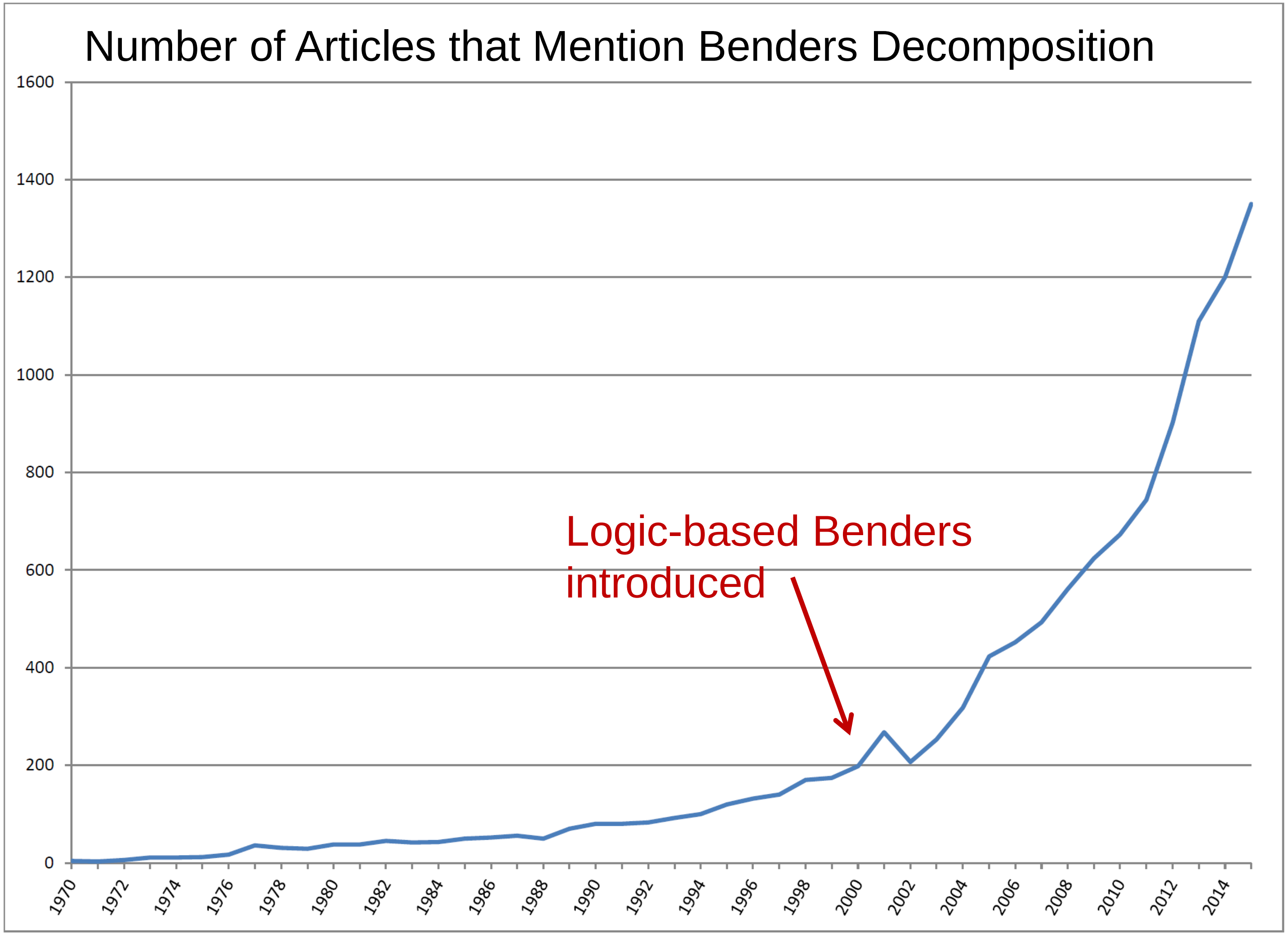
- But classical Benders decomposition has **a serious limitation.**
  - The subproblems must be **linear programming** problems.
    - Or continuous nonlinear programming problems.
    - The **linear programming dual** provides the Benders cuts.



Logic-Based Benders

- **Logic-based Benders decomposition** attempts to overcome this limitation.
  - The subproblems can, in principle, be **any kind of optimization problem**.
    - The Benders cuts are obtained from an **inference dual**.
  - Speedup over state of the art can be **several orders of magnitude**.
  - Yet the Benders cuts must be designed specifically for every class of problems.

Logic-Based Benders



Source: Google Scholar

Logic-Based Benders

- Logic-based Benders decomposition solves a problem of the form

$$\min f(x,y)$$
$$(x,y) \in S$$
$$x \in D_x, y \in D_y$$

  - Where the problem simplifies when ***x* is fixed** to a specific value.

Logic-Based Benders

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

Master problem

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

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)

$x \in D_x$

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

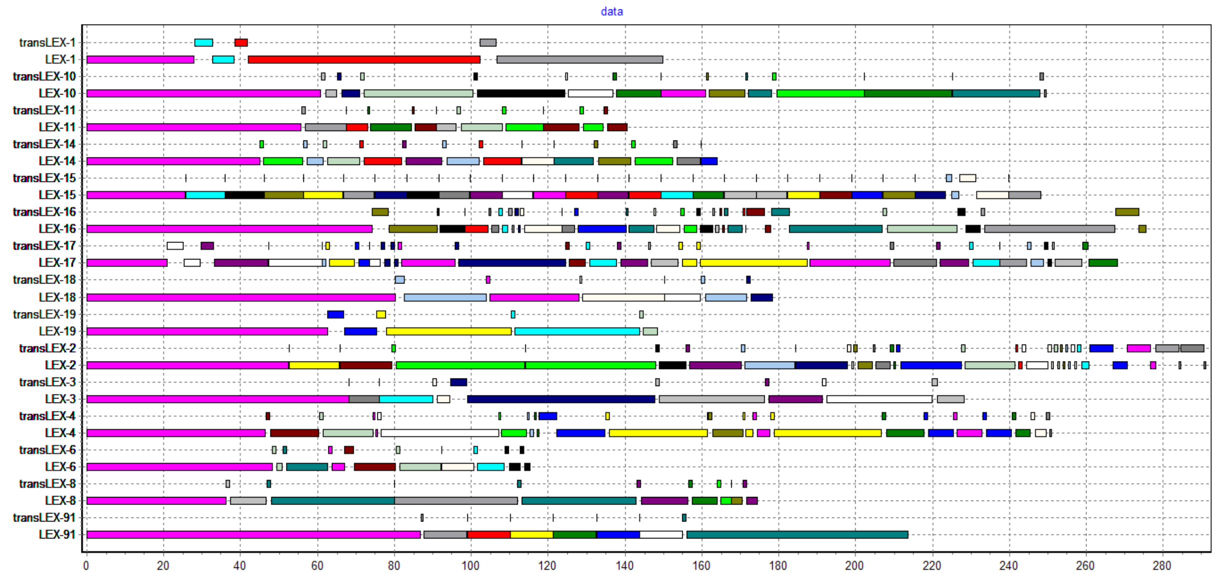
$\min f(\bar{x}, y)$

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

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

Machine Scheduling

- Assign tasks to machines.
- Then schedule tasks assigned to each machine.
  - Subject to time windows.
  - Cumulative scheduling: several tasks can run simultaneously, subject to resource limits.
  - Scheduling problem **decouples** into a separate problem for each machine.



Machine 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.

Machine 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**.

Machine 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.

Machine 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

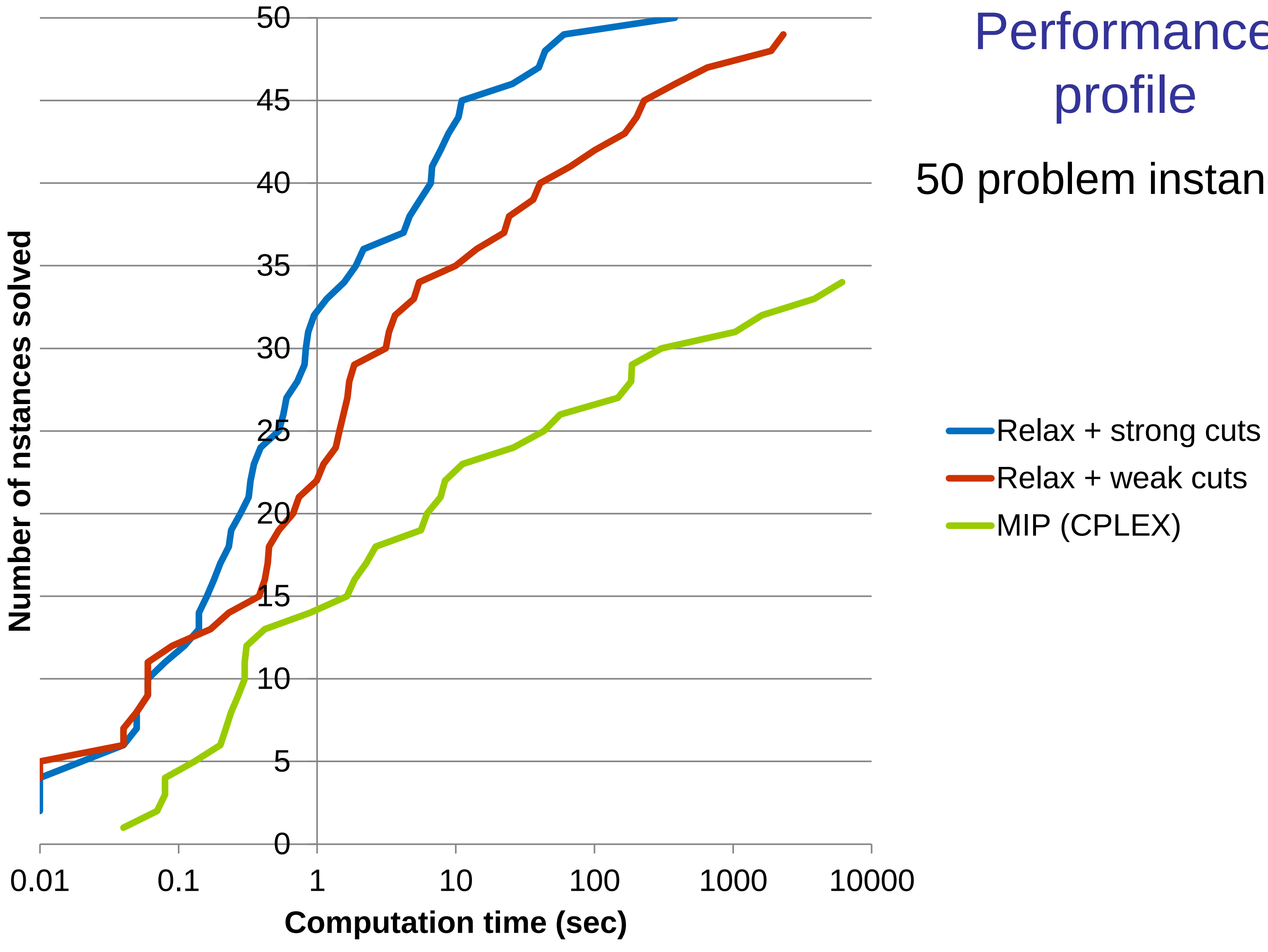
Schedule jobs on each resource.

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.

Performance profile

50 problem instances



Logic-Based Benders Applications

- Planning and scheduling:
  - Machine allocation and scheduling
  - Steel production scheduling
  - Chemical batch processing (BASF, etc.)
  - Auto assembly line management (Peugeot-Citroën)
  - Allocation and scheduling of multicore processors (IBM, Toshiba, Sony)
  - Worker assignment in a queuing environment

Logic-Based Benders Applications

- Other scheduling
  - Lock scheduling
  - Shift scheduling
  - Permutation flow shop scheduling with time lags
  - Resource-constrained scheduling
  - Hospital scheduling
  - Optimal control of dynamical systems
  - Sports scheduling

Logic-Based Benders Applications

- Routing and scheduling
  - Vehicle routing
  - Home health care
  - Food distribution
  - Automated guided vehicles in flexible manufacturing
  - Traffic diversion around blocked routes
  - Concrete delivery

Logic-Based Benders Applications

- Location and Design
  - Wireless local area network design
  - Facility location-allocation
  - Stochastic facility location and fleet management
  - Capacity and distance-constrained plant location
  - Queuing design and control

Logic-Based Benders Applications

- Other
  - Logical inference
  - Logic circuit verification
  - Bicycle sharing
  - Service restoration in a network
  - Inventory management
  - Supply chain management
  - Space packing

Logical Inference

- A fundamental problem in the information age.
  - Can use **SAT solvers** or **logic-based Benders** to deduce facts from a knowledge base.

Logical Inference

- Draw inferences from a clause set
  - Infer everything we can about propositions x_1, x_2, x_3

We can deduce

$$x_1 \vee x_2$$

$$x_1 \vee x_3$$

This is a **projection** onto x_1, x_2, x_3

$$
\begin{array}{llllll}
x_1 & & & \vee\, x_4 & \vee\, x_5 & \\
x_1 & & & \vee\, x_4 & \vee\, \bar{x}_5 & \\
x_1 & & & & \vee\, x_5 & \vee\, x_6 \\
x_1 & & & & \vee\, x_5 & \vee\, \bar{x}_6 \\
 & x_2 & & & \vee\, \bar{x}_5 & \vee\, x_6 \\
 & x_2 & & & \vee\, \bar{x}_5 & \vee\, \bar{x}_6 \\
 & & x_3 & \vee\, \bar{x}_4 & \vee\, x_5 & \\
 & & x_3 & \vee\, \bar{x}_4 & \vee\, \bar{x}_5 &
\end{array}
$$

Logical Inference

- Benders decomposition computes a **projection!**
  - Benders cuts describe projection onto master problem variables.

Current
Master problem

$$\boxed{x_1 \vee x_2}$$

Benders cut
from previous
iteration

Logical Inference

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.

Current Master problem

$x_1 \vee x_2$

solution of master

$(x_1,x_2,x_3) = (F,T,F)$

Resulting subproblem

$$x_4 \vee x_5$$
$$x_4 \vee \bar{x}_5$$
$$x_5 \vee x_6$$
$$x_5 \vee \bar{x}_6$$
$$\bar{x}_4 \vee x_5$$
$$\bar{x}_4 \vee \bar{x}_5$$

Logical Inference

- Benders decomposition computes a **projection!**
  - Benders cuts describe projection onto master problem variables.

Current
Master problem

$x_1 \vee x_2$

solution of master
$(x_1,x_2,x_3) = (F,T,F)$

Resulting
subproblem

$$
\begin{array}{l}
x_4 \vee x_5 \\
x_4 \vee \bar{x}_5 \\
\qquad x_5 \vee x_6 \\
\qquad x_5 \vee \bar{x}_6 \\
\bar{x}_4 \vee x_5 \\
\bar{x}_4 \vee \bar{x}_5
\end{array}
$$

Subproblem is
infeasible.
$(x_1,x_3)=(F,F)$
creates infeasibility

Logical Inference

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.

Current
Master problem

$$x_1 \vee x_2$$
$$x_1 \vee x_3$$

solution of master
$(x_1,x_2,x_3) = (F,T,F)$

Benders cut
(nogood)

Resulting
subproblem

$$x_4 \vee x_5$$
$$x_4 \vee \bar{x}_5$$
$$x_5 \vee x_6$$
$$x_5 \vee \bar{x}_6$$
$$\bar{x}_4 \vee x_5$$
$$\bar{x}_4 \vee \bar{x}_5$$

Subproblem is
infeasible.
$(x_1,x_3)=(F,F)$
creates infeasibility

Logical Inference

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.

Current Master problem

$$x_1 \vee x_2$$
$$x_1 \vee x_3$$

$\rightarrow$ solution of master $(x_1,x_2,x_3) = (F,T,T)$ $\rightarrow$

Resulting subproblem

$$x_4 \vee x_5$$
$$x_4 \vee \bar{x}_5$$
$$x_5 \vee x_6$$
$$x_5 \vee \bar{x}_6$$

Logical Inference

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.

Current Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

solution of master $(x_1,x_2,x_3) = (F,T,T)$

Resulting subproblem

$x_4 \vee x_5$

$x_4 \vee \bar{x}_5$

$x_5 \vee x_6$

$x_5 \vee \bar{x}_6$

Subproblem is **feasible**

Logical Inference

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.

Current
Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

$x_1 \vee \bar{x}_2 \vee \bar{x}_3$

solution of master
$(x_1,x_2,x_3) = (F,T,T)$

Enumerative
Benders cut

Resulting
subproblem

$x_4 \vee x_5$

$x_4 \vee \bar{x}_5$

$x_5 \vee x_6$

$x_5 \vee \bar{x}_6$

Subproblem is
feasible

Logical Inference

- Benders decomposition computes a **projection**!
  - Benders cuts describe projection onto master problem variables.

Current Master problem

$x_1 \vee x_2$

$x_1 \vee x_3$

$x_1 \vee \bar{x}_2 \vee \bar{x}_3$

solution of master $(x_1,x_2,x_3) = (F,T,T)$

Enumerative Benders cut

Resulting subproblem

$x_4 \vee x_5$

$x_4 \vee \bar{x}_5$

$x_5 \vee x_6$

$x_5 \vee \bar{x}_6$

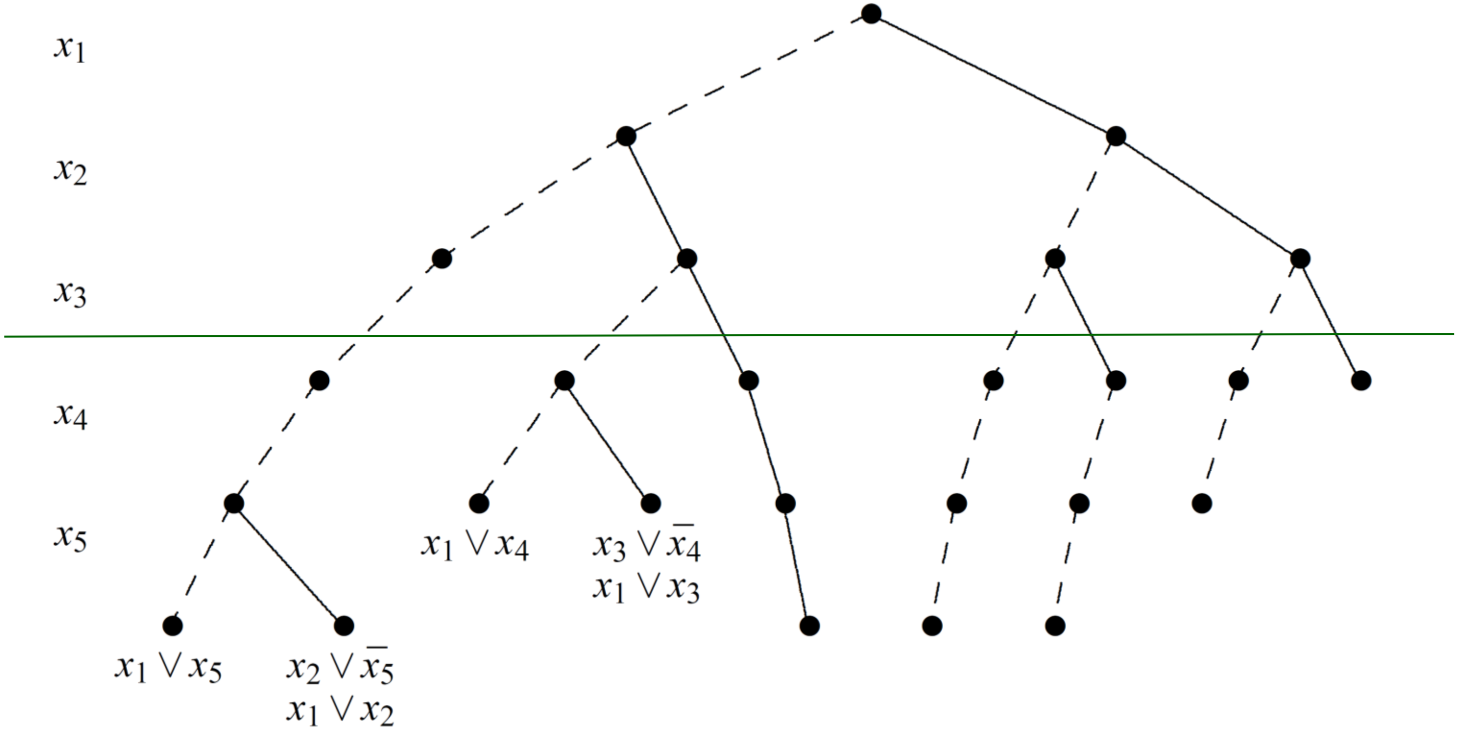
Continue until **master** is infeasible. Black Benders cuts describe projection.

Logical Inference

- **Satisfiability methods** solve the problem by generating **Benders cuts!**
  - Conflict clauses = Benders cuts

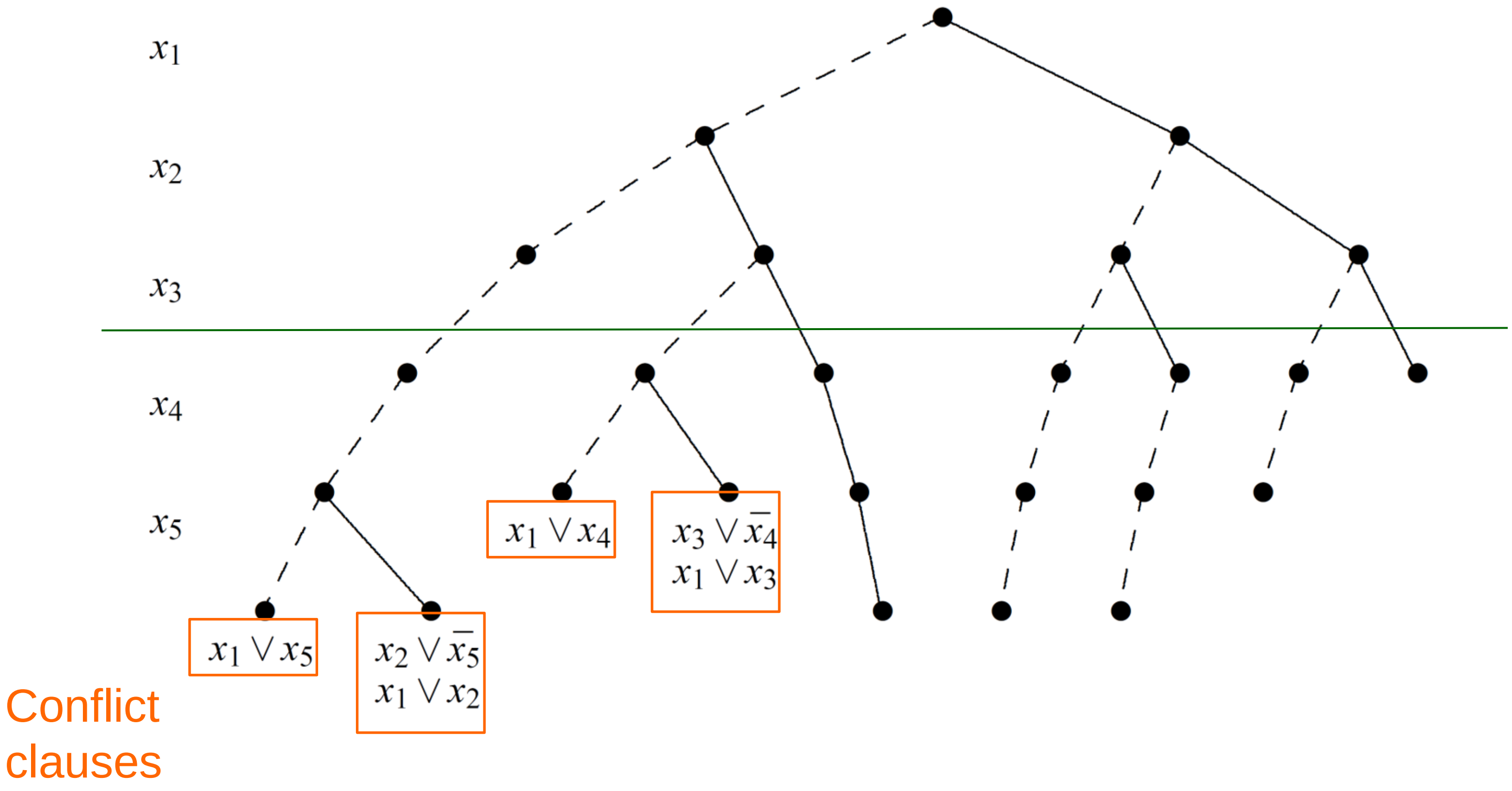
Logical Inference

- Benders cuts = conflict clauses in a SAT algorithm
  – Branch on x_1, x_2, x_3 first.



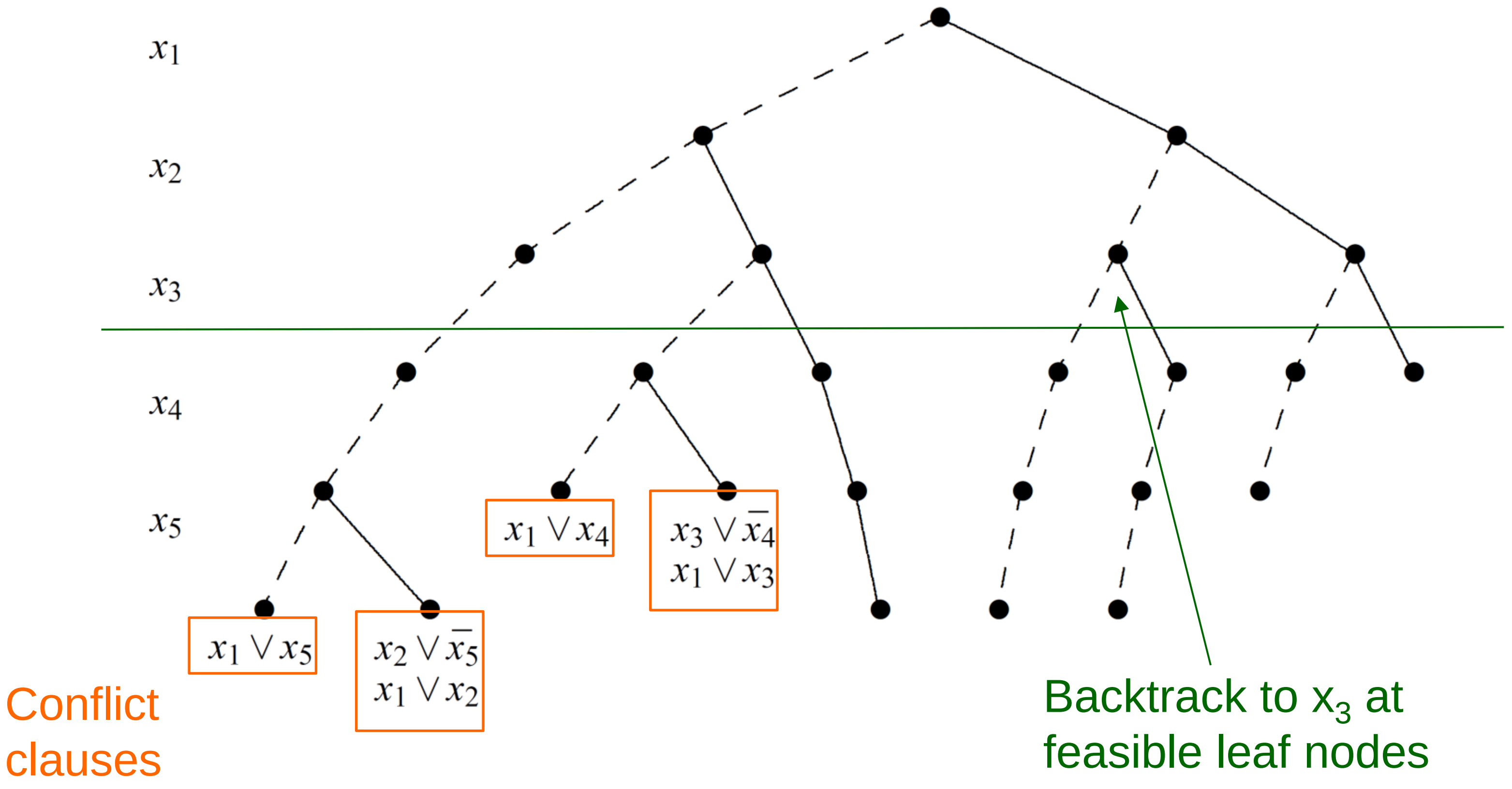
Logical Inference

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



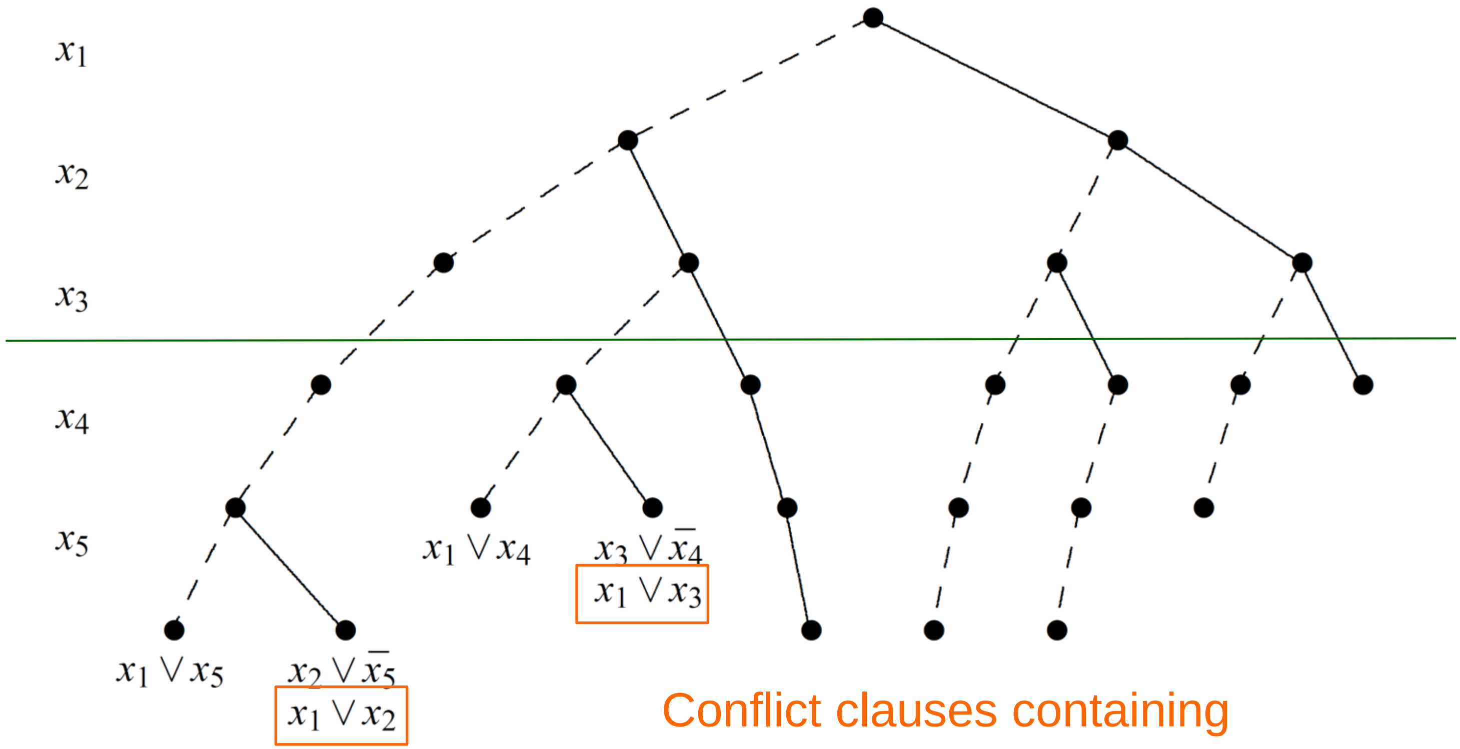
Logical Inference

- Benders cuts = conflict clauses in a SAT algorithm
  – Branch on x_1, x_2, x_3 first.



Logical Inference

- Benders cuts = conflict clauses in a SAT algorithm
  - Branch on x_1, x_2, x_3 first.



Conflict clauses containing
x_1, x_2, x_3 describe projection

Home Health Care

- General home health care problem.
  - Assign **aides** to homebound **patients**.
    - …subject to constraints on aide qualifications and patent preferences.
    - One patient may require a team of aides.
  - **Route** each aide through assigned patients, observing **time windows**.
    - …subject to constraints on hours, breaks, etc.

Home Health Care

- A large industry, and **rapidly growing**.
  - Roughly as large as all courier and delivery services.

Projected Growth
of Home Health Care Industry

	2014	2018
U.S. revenues, $ billions	75	150
World revenues, $ billions	196	306

Increase in U.S. Employment, 2010-2020

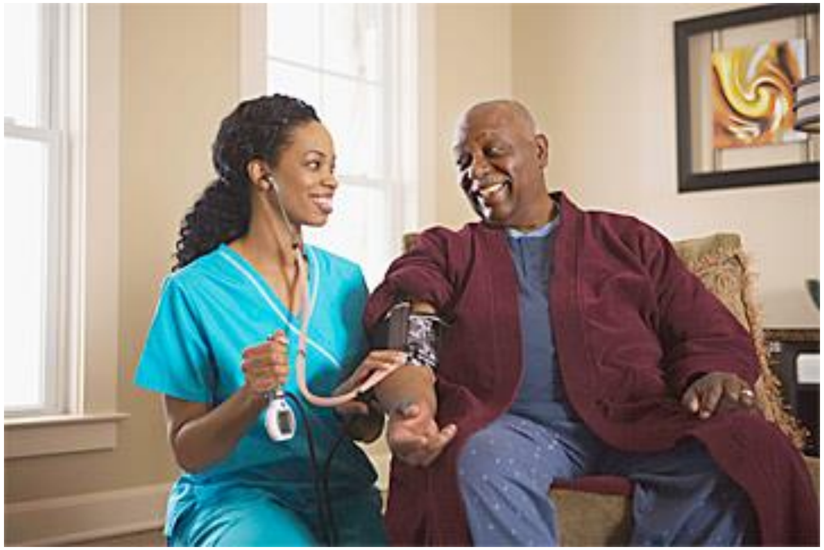
Home health care industry	70%
Entire economy	14%

Home Health Care

- Advantages of home health care
  - Lower cost
    - Hospital & nursing home care is very expensive.
  - No hospital-acquired infections
    - Less exposure to superbugs.
  - Preferred by patients
    - Comfortable, familiar surroundings of home.
    - Sense of control over one’s life.
  - Supported by new equipment & technology
    - IT integration with hospital systems.
    - Online consulting with specialists.

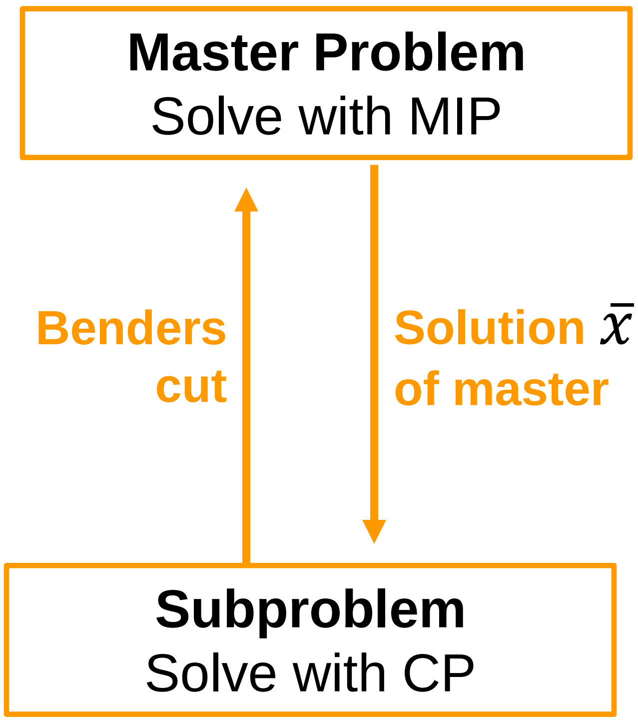
Home Health Care

- Critical factor to realize cost savings:
  - Aides must be **efficiently** scheduled.
- This is our task.
  - Focus on home hospice care.
  - Rolling schedule – update as patient population evolves



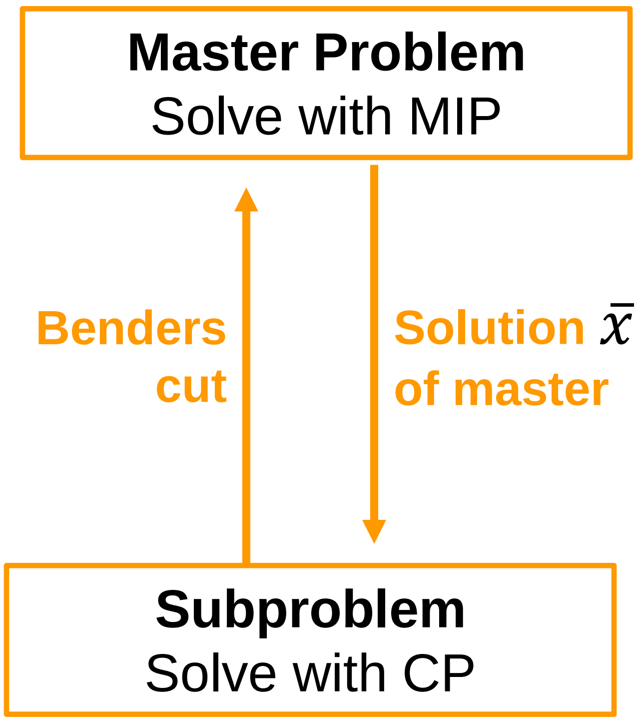
Home Health Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.



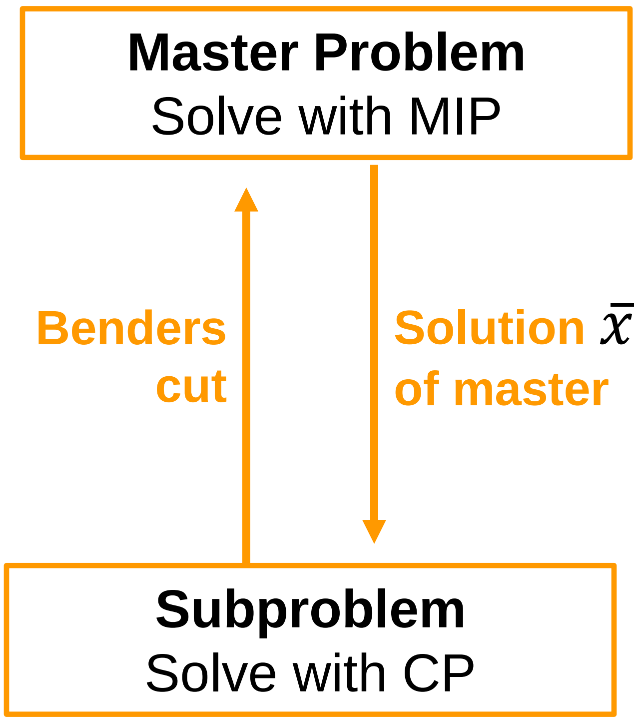
Home Health Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.
  - **Schedule home visits** in subproblem.
    - Cyclic weekly schedule.
    - No visits on weekends.



Home Health Care

- Solve with Benders decomposition.
  - **Assign aides to patients** in master problem.
    - Maximize number of patients served by a given set of aides.
  - **Schedule home visits** in subproblem.
    - Cyclic weekly schedule.
    - No visits on weekends.
  - Subproblem **decouples** into a scheduling problem for each aide and each day of the week.



Home Health Care

= 1 if patient *j* scheduled

= 1 if patient *j* assigned to aide *i*

$$\max \sum_j \boxed{\delta_j}$$

$$\sum_i \boxed{x_{ij}} = \delta_j, \quad \text{all } j$$

= 1 if patient *j* assigned to aide *i* on day *k*

Required number of visits per week

$$\sum_{i,k} \boxed{y_{ijk}} = \boxed{v_j}\delta_j, \quad \text{all } j$$

$$y_{ijk} \le x_{ij}, \quad \text{all } i, j, k$$

Spacing constraints on visit days

Benders cuts

Relaxation of subproblem

$$\delta_j, x_{ij}, y_{ijk} \in \{0, 1\}$$

Master Problem

Home Health Care

- For a rolling schedule:
  - Schedule **new patients**, drop **departing patients** from schedule.
    - Provide continuity for remaining patients as follows:
  - Old patients served by **same aide** on **same days**.
    - Fix $y_{ijk} = 1$ for the relevant aides, patients, and days.

Home Health Care

Scheduling subproblem for aide *i*, day *k*

*n*th patient in sequence

Set of patients assigned to aide *i*, day *k*

$$\text{alldiff}\{\boxed{\pi_n} \mid n = 1, \ldots, |P_{ik}|\}$$

start time

$$[s_j, s_j + p_j] \subseteq [r_j, d_j], \quad \text{all } j \in P_{ik}$$

$$\boxed{s_{\pi_n}} + \boxed{p_{\pi_n}} + \boxed{t_{\pi_n \pi_{n+1}}} \leq s_{\pi_{n+1}}, \quad n = 1, \ldots, |P_{ik}| - 1$$

Visit duration

Travel time

Modeled with interval variables in CP solver.

Home Health Care

- Benders cuts.
  - Find a small set of patients that create infeasibility…
    - …by re-solving the each infeasible scheduling problem repeatedly.

$$\sum_{j \in \boxed{\bar{P}_{ik}}} (1 - y_{ijk}) \geq 1$$

Reduced set of patients whose assignment to aide *i* on day *k* creates infeasibility

Home Health Care

- Auxiliary cuts based on symmetries.
  - A cut for valid for aide *i*, day *k* is also valid for aide *i* on other days.
    - This gives rise to a large number of cuts.
  - The auxiliary cuts can be summed without sacrificing optimality.
    - Original cut ensures convergence to optimum.
    - This yields 2 cuts per aide:

$$\sum_{j \in \bar{P}_{ik}} (1 - y_{ijk}) \geq 1$$

$$\sum_{k \neq k} \sum_{j \in \bar{P}_{ik}} (1 - y_{ijk'}) \geq 4$$

Home Health Care

- Include relaxation of subproblem in the master problem.
  - Necessary for good performance.
  - Use **time window relaxation** for each scheduling problem.
  - Simplest relaxation for aide *i* and day *k*:

$$\sum_{j \in \boxed{J(a,b)}} p_j y_{ijk} \leq b - a$$

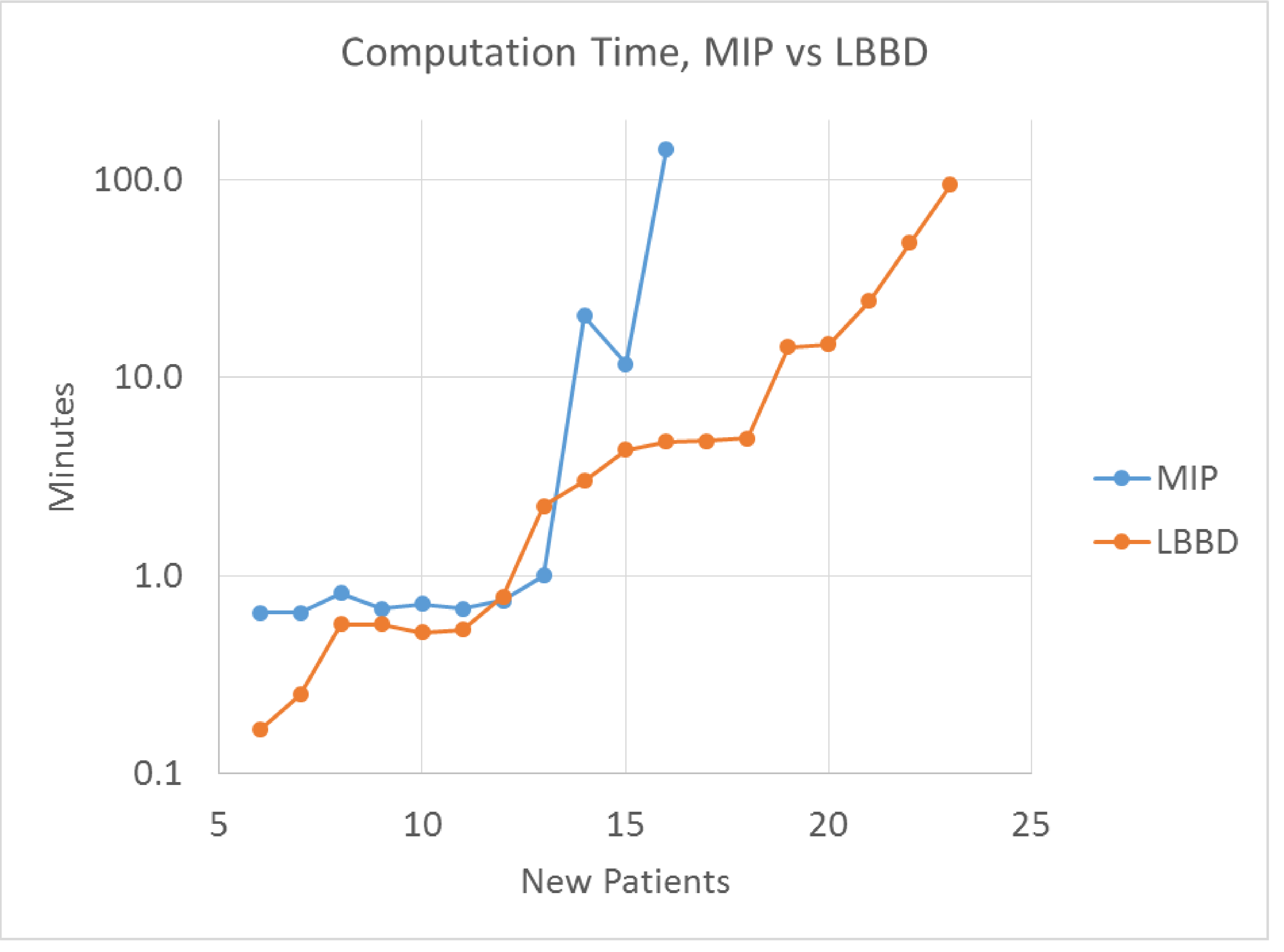
Set of patients whose time window fits in interval [*a*, *b*].

Can use several intervals.

Home Health Care

- Improved relaxation.
  - **Basic idea:** Augment visit duration p_j with travel time to (or from) location j from **closest** patient or aide home base.

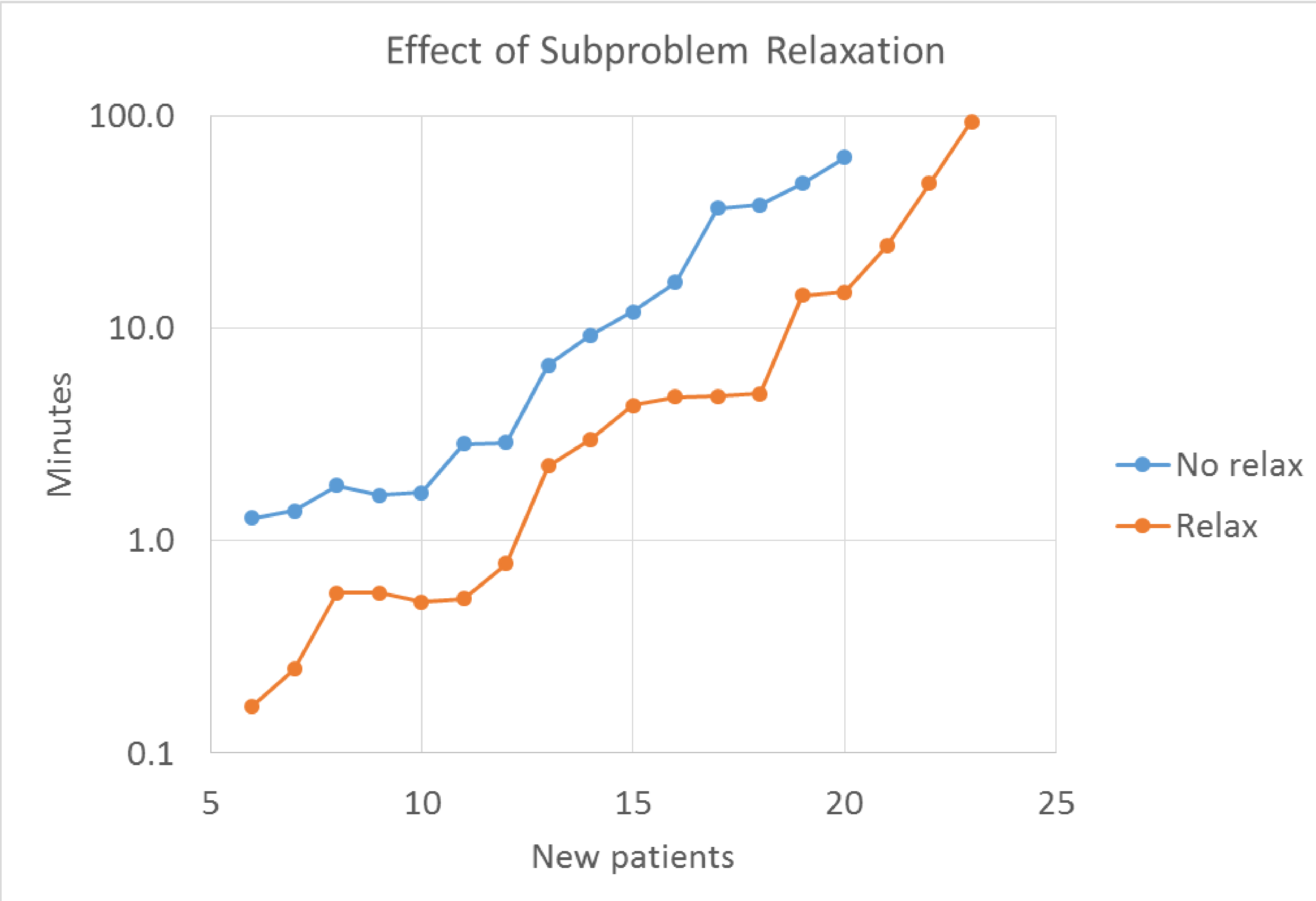
Home Health Care



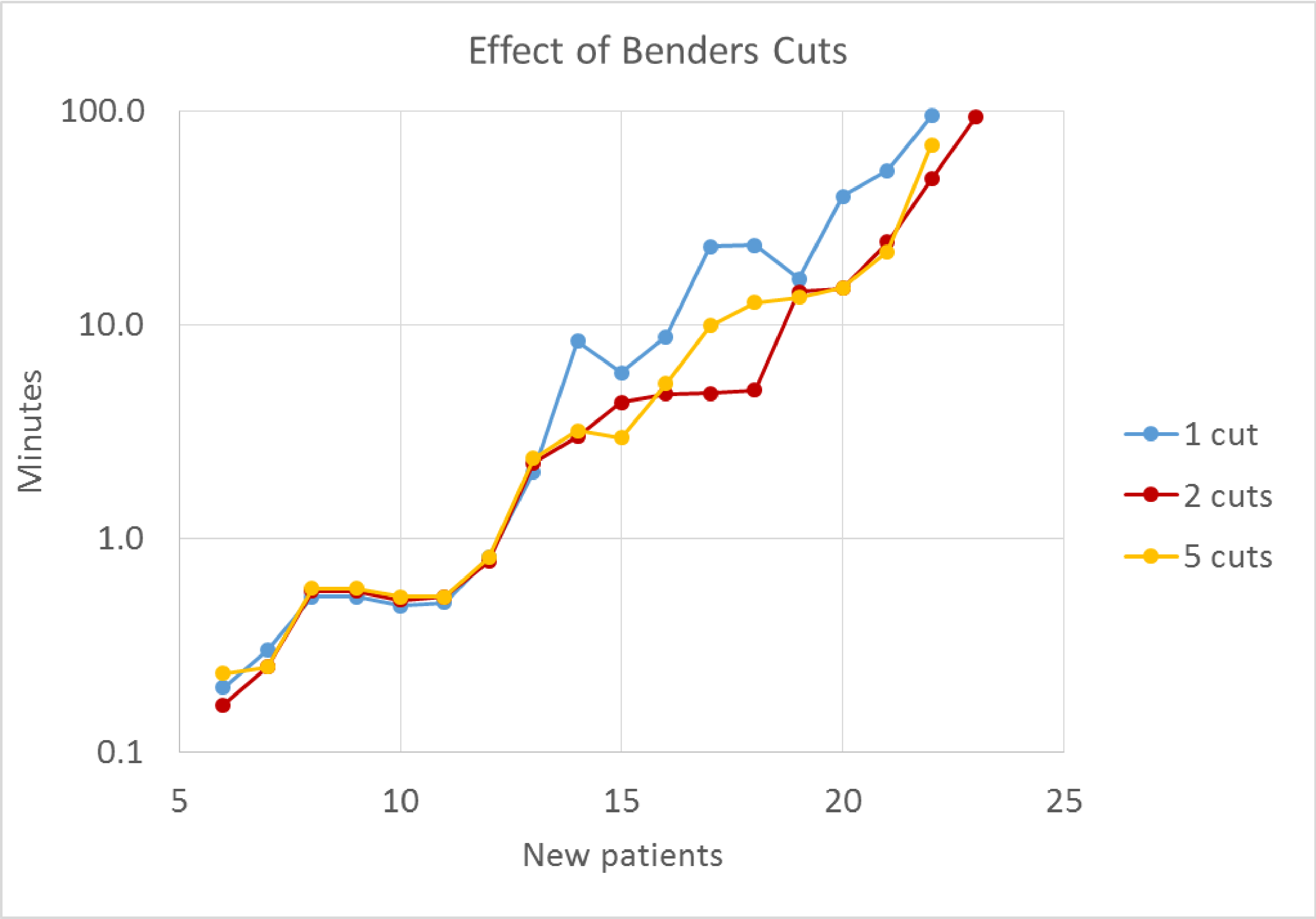
Home Health Care

- Practical implications
  - LBBD **scales up** to realistic size
    - One month advance planning in 60 patient population
    - Assuming 5-8% weekly turnover
  - Advantage of **exact** solution method
    - We know **for sure** whether existing staff will cover projected demand.

Home Health Care



Home Health Care



References

Applications of Logic-Based Benders Decomposition

Benders decomposition [7] was introduced in 1962 to solve applications that become linear programming (LP) problems when certain *search variables* are fixed. "Generalized" Benders decomposition, proposed by Geoffrion in 1972 [25], extended the method to nonlinear programming subproblems.

Logic-based Benders decomposition (LBBD) allows the subproblem to be any optimization problem. LBBD was introduced in [32], formally developed in 2000 [33], and tested computationally in [39]. *Branch and check* is introduced in [33] and tested computationally in [69]. *Combinatorial Benders cuts* for mixed integer programming are proposed in [18].

One of the first applications [43] was a planning and scheduling problem. Updated experiments [17] show that LBBD is orders of magnitude faster than state-of-the-art MIP, with the advantage over CP even greater). Similar results have been obtained for various planning and scheduling problems [15, 21, 30, 34, 35, 37, 71].

Other successful applications of LBBD include steel production scheduling [29], inventory management [74], concrete delivery [44], shop scheduling [3, 13, 27, 28, 59], hospital scheduling [57], batch scheduling in chemical plants [49, 70], computer processor scheduling [8, 9, 12, 22, 31, 46, 47, 48, 58, 62], logic circuit verification [40], shift scheduling [5, 60], lock scheduling [73], facility location [23, 66], space packing [20, 50], vehicle routing [19, 51, 53, 56, 61, 75], bicycle sharing [45], network design [24, 52, 63, 65], home health care [16], service restoration [26], supply chain management [68], food distribution [64], queuing design and control [67], optimal control of dynamical systems [11], propositional satisfiability [1], quadratic programming [2, 41, 42], chordal completion [10], and sports scheduling [14, 54, 55, 72]. LBBD is compared with branch and check in [6]. It is implemented in the general-purpose solver SIMPL [76].

References

[1] F. Bacchus, S. Dalmao, and T. Pitassi. Relaxation search: A simple way of managing optional clauses. In *AAAI Conference on Artificial Intelligence*. 2014.

[2] L. Bai, J. E. Mitchell, and J.-S. Pang. On convex quadratic programs with linear complementarity constraints. *Computational Optimization and Applications*, 54:517–554, 2012.

[3] M. A. Bajestani and J. C. Beck. Scheduling a dynamic aircraft repair shop with limited repair resources. *Journal of Artificial Intelligence Research*, 47:35–70, 2013.

[4] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, Dordrecht, 2001.

[5] A. Y. Barlatt, A. M. Cohn, and O. Gusikhin. A hybridization of mathematical programming and dominance-driven enumeration for solving shift-selection and task-sequencing problems. *Computers and Operations Research*, 37:1298–1307, 2010.

[6] J. C. Beck. Checking up on branch-and-check. In D. Cohen, editor, *Principle and Practice of Constraint Programming (CP)*, volume 6308 of *Lecture Notes in Computer Science*, pages 84–98, 2010.

[7] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.

[8] L. Benini, D. Bertozzi, A. Guerri, and M. Milano. Allocation and scheduling for MPSoCs via decomposition and no-good generation. In *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2005.

[9] L. Benini, M. Lombardi, M. Mantovani, M. Milano, and M. Ruggiero. Multi-stage Benders decomposition for optimizing multicore architectures. In L. Perron and M. A. Trick, editors, *CPAIOR 2008 Proceedings*, volume 5015 of *Lecture Notes in Computer Science*, pages 36–50. Springer, 2008.

[10] D. Bergman and A. U. Raghunathan. A Benders approach to the minimum chordal completion problem. In L. Michel, editor, *CPAIOR Proceedings*, volume 9075 of *Lecture Notes in Computer Science*, pages 47–64. Springer, 2015.

[11] A. H. Borzabadi and M. E. Sadjadi. Optimal control of hybrid systems by logic-based Benders decomposition. In A. Giua, C. Mahulea, M. Silva, and J. Zaytoon, editors, *Analysis and Design of Hybrid Systems*, volume 3, pages 104–107, 2009.

[12] H. Cambazard, P.-E. Hladik, A.-M. Déplanche, N. Jussien, and Y. Trinquet. Decomposition and learning for a hard real time task allocation problem. In M. Wallace, editor, *Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 153–167. Springer, 2004.

[13] E. Çoban and J. N. Hooker. Single-facility scheduling by logic-based Benders decomposition. *Annals of Operations Research*, 210:245–272, 2013.

[14] K. K. H. Cheung. A Benders approach for computing lower bounds for the mirrored traveling tournament problem. *Discrete Optimization*, 6:189–196, 2009.

[15] Y. Chu and Q. Xia. A hybrid algorithm for a class of resource-constrained scheduling problems. In R. Barták and M. Milano, editors, *CPAIOR 2005 Proceedings*, volume 3524 of *Lecture Notes in Computer Science*, pages 110–124. Springer, 2005.

[16] A. Ciré and J. N. Hooker. A heuristic logic-based Benders method for the home health care problem. Presented at Matheuristics 2012, Angra dos Reis, Brazil, 2012.

[17] A. A. Ciré, E. Çoban, and J. N. Hooker. Mixed integer programming vs logic-based Benders decomposition for planning and scheduling. In C. Gomes and M. Sellmann, editors, *CPAIOR 2013 Proceedings*, pages 325–331, 2013.

[18] G. Codato and M. Fischetti. Combinatorial Benders cuts for mixed-integer linear programming. *Operations Research*, 54:756–766, 2006.

[19] A. I. Corréa, A. Langevin, and L. M. Rousseau. Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–378. Springer, 2004.

[20] J.-F. Côté, M. Dell'Amico, and M. Iori. Combinatorial Benders cuts for the strip packing problem. *Operations Research*, 62:643–661, 2014.

[21] T. O. Davies, A. R. Pearce, P. J. Stuckey, and N. Lipovetzky. Sequencing operator counts. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 61–69, 2015.

[22] A. Emeretlis, G. Theodoridis, P. Alefragis, and N. Voros. Mapping DAGs on heterogeneous platforms using logic-based Benders decompostion. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 119–124. IEEE, 2015.

[23] M. M. Fazel-Zarandi and J. C. Beck. Solving a location-allocation problem with logic-based Benders decomposition. In I. P. Gent, editor, *Principles and Practice of Constraint Programming (CP 2009)*, volume 5732 of *Lecture Notes in Computer Science*, pages 344–351, New York, 2009. Springer.

[24] B. Gendron, R. G. Garroppo, G. Nencioni, M. G. Scutellà, and L. Tavanti. Benders decomposition for a location-design problem in green wireless local area networks. *Electronic Notes in Discrete Mathematics*, 41:367–374, 2013.

[25] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.

[26] J. Gong, E. E. Lee, J. E. Mitchell, and W. A. Wallace. Logic-based multiobjective optimization for restoration planning. In W. Chaovalitwongse, K. C. Furman, and P. M. Pardalos, editors, *Optimization and Logistics Challenges in the Enterprise*, pages 305–324. 2009.

[27] O. Guyon, P. Lemaire, E. Pinson, and D. Rivreau. Solving an integrated job-shop problem with human resource constraints. *Annals of Operations Research*, 213:147–171, 2014.

[28] I. Hamdi and T. Loukil. Logic-based Benders decomposition to solve the permutation flowshop scheduling problem with time lags. In *International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, pages 1–7. IEEE, 2013.

[29] I. Harjunkoski and I. E. Grossmann. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering*, 25:1647–1660, 2001.

[30] I. Harjunkoski and I. E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26:1533–1552, 2002.

[31] P.-E. Hladik, H. Cambazard, A.-M. Déplanche, and N. Jussien. Solving a real-time allocation problem with constraint programming. *Journal of Systems and Software*, 81:132—149, 2008.

[32] J. N. Hooker. Logic-based Benders decomposition. In *INFORMS National Meeting (INFORMS 1995)*, 1995.

[33] J. N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.

[34] J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10:385–401, 2005.

[35] J. N. Hooker. An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11:139–157, 2006.

[36] J. N. Hooker. *Integrated Methods for Optimization*. Springer, 2007.

[37] J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55:588–602, 2007.

[38] J. N. Hooker. *Integrated Methods for Optimization, 2nd ed.* Springer, 2012.

[39] J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96:33–60, 2003.

[40] J. N. Hooker and H. Yan. Logic circuit verification by Benders decomposition. In V. Saraswat and P. Van Hentenryck, editors, *Principles and Practice of Constraint Programming: The Newport Papers*, pages 267–288, Cambridge, MA, 1995. MIT Press.

[41] J. Hu, J. E. Mitchell, and J.-S. Pang. An LPCC approach to nonconvex quadratic programs. *Mathematical Programming*, 133:243–277, 2012.

[42] J. Hu, J. E. Mitchell, J.-S. Pang, K. P. Bennett, and G. Kunapuli. On the global solution of linear programs with linear complementarity constraints. *SIAM Journal on Optimization*, 19:445–471, 2008.

[43] V. Jain and I. E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276, 2001.

[44] J. Kinable and M. Trick. A logic-based Benders approach to the concrete delivery problem. In H. Simonis, editor, *CPAIOR Proceedings*, volume 8451 of *Lecture Notes in Computer Science*, pages 176–192. Springer, 2014.

[45] C. Kloimüllner, P. Papazek, B. Hu, and G. R. Raidl. A cluster-first route-second approach for balancing bicycle sharing systems. In *International Conference on Computer Aided Systems Theory (EUROCAST)*, volume 9520 of *Lecture Notes in Computer Science*, pages 439–446. Springer, 2015.

[46] W. Liu, Z. Gu, J. Xu, X. Wu, and Y. Ye. Satisfiability modulo graph theory for task mapping and scheduling on multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 22:1382–1389, 2011.

[47] W. Liu, M. Yuan, X. He, Z. Gu, and X. Liu. Efficient SAT-based mapping and scheduling of homogeneous synchronous dataflow graphs for throughput optimization. In *Real-Time Systems Symposium*, pages 492–504. IEEE, 2008.

[48] M. Lombardi, M. Milano, M. Ruggiero, and L. Benini. Stochastic allocation and scheduling for conditional task graphs in multi-processor systems-on-chip. *Journal of Scheduling*, 13:315–345, 2010.

[49] C. T. Maravelias and I. E. Grossmann. Using MILP and CP for the scheduling of batch chemical processes. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

[50] J. Maschler and G. Raidl. Logic-based Benders decomposition for the 3-staged strip packing problem. In *International Conference on Operations Research (German OR Society)*, 2015.

[51] T. Nishi, Y. Hiranaka, and I. E. Grossmann. A bilevel decomposition algorithm for simultaneous production scheduling and conflict-free routing for automated guided vehicles. *Computers and Operations Research*, 38:876–888, 2011.

[52] B. Peterson and M. Trick. A Benders' approach to a transportation network design problem. In W.-J. van Hoeve and J. N. Hooker, editors, *CPAIOR 2009 Proceedings*, volume 5547 of *Lecture Notes in Computer Science*, pages 326–327, New York, 2009. Springer.

[53] G. R. Raidl, T. Baumhauer, and B. Hu. Speeding up logic-based Benders decomposition by a metaheuristic for a bi-level capacitated vehicle routing problem. In *International Workshop on Hybrid Metaheuristics*, volume 8457 of *Lecture Notes in Computer Science*, pages 183–197. Springer, 2014.

[54] R. V. Rasmussen. Scheduling a triple round robin tournament for the best Danish soccer league. *European Journal of Operational Research*, 20:795–810, 2008.

[55] R. V. Rasmussen and M. A. Trick. A Benders approach to the constrained minimum break problem. *European Journal of Operational Research*, 177:198–213, 2007.

[56] S. Riazi, C. Seatzu, O. Wigstrom, and B. Lennartson. Benders/gossip methods for heterogeneous multi-vehicle routing problems. In *IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, pages 1–6, 2013.

[57] V. Roshanaei, D. M. Aleman, and D. Urbach. Logic-based Benders decomposition approaches with application to operating room scheduling. In *INFORMS National Meeting*, 2015.

[58] M. Ruggiero, A. Guerri, D. Bertozzi, F. Poletti, and M. Milano. Communication-aware allocation and scheduling framework for stream-oriented multi-processor systems-on-chip. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 3–8. European Design and Automation Association, 2006.

[59] R. Sadykov. A hybrid branch-and-cut algorithm for the one-machine scheduling problem. In J. C. Régin and M. Rueher, editors, *CPAIOR Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 409–415. Springer, 2004.

[60] D. Salvagnin and T. Walsh. A hybrid MIP/CP approach for multi-activity shift scheduling. In M. Milano, editor, *Principles and Practice of Constraint Programming*, volume 7514 of *Lecture Notes in Computer Science*, pages 633–646. Springer, 2012.

[62] N. Satish, K. Ravindran, and K. Keutzer. A decomposition-based constraint optimization approach for statically scheduling task graphs with communication delays to multiprocessors. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 57–62. EDA Consortium, 2007.

[63] S. Shen and J. C. Smith. A decomposition approach for solving a broadcast domination network design problem. *Annals of Operations Research*, 210:333–360, 2011.

[64] S. Solak, C. Scherrer, and A. Ghoniem. The stop-and-drop problem in nonprofit food distribution networks. *Annals of Operations Research*, 221:407–426, 2014.

[65] Z. C. Taşkın, J. C. Smith, S. Ahmed, and A. J. Schaefer. Cutting plane algorithms for solving a stochastic edge-partition problem. *Discrete Optimization*, 6:420–435, 2009.

[66] S. Tarim, S. Armagan, and I. Miguel. A hybrid Benders decomposition method for solving stochastic constraint programs with linear recourse. In B. Hnich, M. Carlsson, F. Fages, and F. Rossi, editors, *International Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)*, pages 133–148. Springer, 2006.

[67] D. Terekhov, J. C. Beck, and K. N. Brown. Solving a stochastic queueing design and control problem with constraint programming. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007)*, volume 1, pages 261–266. AAAI Press, 2007.

[68] D. Terekhov, M. K. Doğru, U. Özen, and J. C. Beck. Solving two-machine assembly scheduling problems with inventory constraints. *Computers and Industrial Engineering*, 63:120–134, 2012.

[69] E. Thorsteinsson. Branch and check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh, editor, *Principles and Practice of Constraint Programming (CP 2001)*, volume 2239 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2001.

[70] C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24:431–448, 2002.

[71] T. T. Tran and J. C. Beck. Logic-based Benders decomposition for alternative resource scheduling with sequence dependent setups. In *European Conference on Artificial Intelligence (ECAI)*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 774–779. IOS Press, 2012.

[72] M. Trick and H. Yildiz. Benders cuts guided large neighborhood search for the traveling umpire problem. In P. Van Hentenryck and L. Wolsey, editors, *CPAIOR Proceedings*, volume 4510 of *Lecture Notes in Computer Science*, pages 332–345. Springer, 2007.

[73] J. Verstichel, J. Kinable, P. De Causmaecker, and G. Vanden Berghe. A combinatorial Benders decomposition for the lock scheduling problem. *Computers and Operations Research*, 54:117–128, 2015.

[74] D. Wheatley, F. Gzara, and E. Jewkes. Logic-based Benders decomposition for an inventory-location problem with service constraints. *Omega*, 55:10–23, 2015.

[75] Q. Xia, A. Eremin, and M. Wallace. Problem decomposition for traffic diversions. In J. C. Régin and M. Rueher, editors, *CPAIOR 2004 Proceedings*, volume 3011 of *Lecture Notes in Computer Science*, pages 348–363. Springer, 2004.

[76] T. H. Yunes, I. Aron, and J. N. Hooker. An integrated solver for optimization problems. *Operations Research*, 58:342–356, 2010.

Future of AI

- Superintelligence + autonomy = a threat?
  - How do we formulate ethics for a machine?
  - Or for humans!
  - We are making progress: analytical normative ethics.
- Do autonomous machines have rights and duties?
  - Do you have to be nice to your robot?
  - A truly autonomous machine is ethical!