Integrating Solution Methods in a Modeling Language

John Hooker

Carnegie Mellon University

OR41, London, September 1999

Joint work with…

Hak-Jin Kim, *Carnegie Mellon University*
Greger Ottosson, *Uppsala Universitet*
Erlendur Thorsteinsson, *Carnegie Mellon University*

Integration of optimization and constraint satisfaction

- Optimization and constraint satisfaction have complementary strengths.
- There is much interest in combining them, but their different origins have impeded unification.
  - Optimization -- operations research, mathematics
  - Constraint satisfaction -- artificial intelligence, computer science
- This barrier is now being overcome, but there is no generally accepted scheme for unification.

Today's Topic

Can one design a modeling language whose syntax indicates how the two approaches can combine to solve the problem?

First… What are the complementary strengths we want to combine?

Complementary strengths

- Problem types.
  - Optimization excels at loosely-constrained problems in which find the best solution is the primary task.
  - Constraint satisfaction is more effective on tightly-constrained problems in which finding a feasible solution is paramount.

Complementary strengths

- Relaxation and inference.
  - Optimization creates strong relaxations with cutting planes, Lagrangean relaxation, etc. These provide bounds on the optimal value.
  - Constraint satisfaction exploits the power of inference, especially in domain reduction algorithms. This reduces the search space.

Complementary strengths

- Exploiting structure
  - Optimization relies on deep analysis of specific *classes of problems*.
  - Constraint satisfaction relies on deep analysis of *subsets of constraints*, within a given problem, that have special structure.

Complementary strengths

- Modeling style
  - The syntax of a modeling language (e.g., linear programming) forces the model to fit the solver's mold. But the language is restrictive.
  - The modeling language is more expressive and may permit succinct, more natural models. But the modeler must identify structure in subsets of the constraints if the problem is to be soluble.

Procedural vs. declarative

- The issue of procedural vs. declarative modeling is orthogonal to the issue of how solution methods can be combined.
  - Optimization models are traditionally declarative.
  - Constraint satisfaction techniques are usually applied to constraint (logic) programs, which are quasi-procedural.

- A mathematical *program* is a program in the sense of a plan (specially, a mathematical model of a plan).
- A constraint *program* is a program in the sense of a computer program.
  - One tells what a desirable plan is *like*.
  - The other tells how to *find* a desirable plan.
- Constraint satisfaction techniques and modeling ideas can be used in declarative models.

•Models considered here are declarative.

• Their syntax will indicate how solutions methods may be combined.

• Procedural or quasi-procedural modeling, such as constraint (logic) programming, can also combine methods.

• But how they might do so is not today’s topic.

Relxation & Inference

Each approach as a distinctive contribution:

Optimizs

Optimization and constraint satisfaction use both search an inference. However, …

- Optimization focuses on *search* and delivers a *specific solution*. If the search is thorough enough or smart enough, the solution is good.
    - *Inference* in the form of cutting planes can make the search smarter.
- Constraint satisfaction focuses on *inference* (mostly in the form of "domain reduction") and *narrows down which values* each variable can have in a good solution. If a single value is isolated for each variable, a good solution is found.
    - If a single value is not isolated, one can *search* over the possibilities.

A simple example will illustrate how these approaches may be combined ...

A motivating example

$$\begin{aligned} \min \quad & 4x_1 + 3x_2 + 5x_3 \\ \text{subject to} \quad & 4x_1 + 2x_2 + 4x_3 \geq 17 \\ & x_1 \neq x_2,\ x_1 \neq x_3,\ x_2 \neq x_3 \\ & x_j \in \{1,\ldots,5\} \end{aligned}$$

Formulate and solve 3 ways:

- a constraint satisfaction problem
- an integer programming problem
- a combined approach

Solve as a constraint satisfaction problem

$$4x_1 + 3x_2 + 5x_3 \leq z$$

$$4x_1 + 2x_2 + 4x_3 \geq 17$$

$$\text{all-different}\{x_1, x_2, x_3\}$$

$$x_j \in \{1, \ldots, 5\}$$

Start with z = ∞.
Will decrease as feasible solutions are found.

Global Constraints

All-different is a *global constraint*.

- It represents a set of constraints with special structure.
- This allows a special-purpose inference procedure to be applied.
- The modeler recognizes the structure.

Domain Reduction

Domain reduction is a special type of logical inference method.

- It infers that each variable can take only certain values.
- That is, it reduces the domains of the variables.
- Ideally it maintains *hyperarc consistency*: every value in any given domain is part of some feasible solution.

Maintain hyperarc consistency on

$$\text{all-different}\{x_1, x_2, x_3\}$$

For example, suppose domains are

$$x_1 \in \{1\,2\quad\}$$
$$x_2 \in \{1\,2\quad\}$$
$$x_3 \in \{1\,2\,3\,4\,5\}$$

After domain reduction,

$$x_1 \in \{1\,2\quad\}$$
$$x_2 \in \{1\,2\quad\}$$
$$x_3 \in \{\quad 3\,4\,5\}$$

In general, apply maximum cardinality bipartite matching + a theorem of Berge

Bounds Consistency

Bounds consistency means that the minimum and maximum element of any given domain are part of some feasible solution.

It is weaker than hyperarc consistency but easier to maintain.

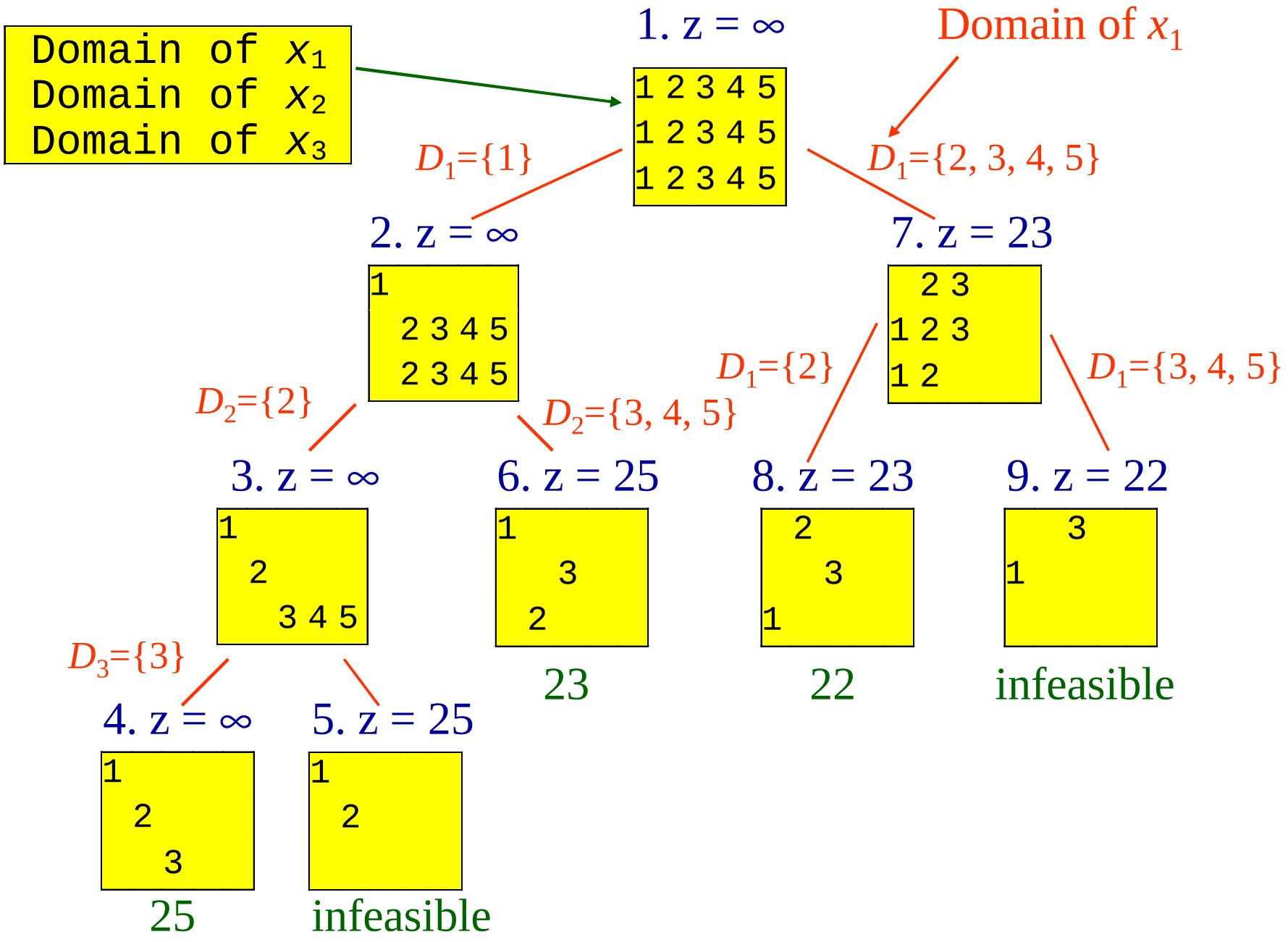
Maintain bounds consistency on

$$4x_1 + 3x_2 + 5x_3 \leq z$$
$$4x_1 + 2x_2 + 4x_3 \geq 17$$

Using the 2nd inequality, for example, modify the variable domains (range of possible values) so that

$$x_1 \geq \left\lceil \frac{17 - 2x_2^{\max} - 4_3^{\max}}{4} \right\rceil$$
$$x_2 \geq \left\lceil \frac{17 - 4x_1^{\max} - 4_3^{\max}}{2} \right\rceil$$
$$x_3 \geq \left\lceil \frac{17 - 4x_1^{\max} - 2_2^{\max}}{4} \right\rceil$$

- Keep cycling through domain reduction procedures until fixed point is reached.
- Do this at every node of a branching tree.
- Branch by domain splitting.


Domain of x_1
Domain of x_2
Domain of x_3
1. z = ∞
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
Domain of x_1
D_1={1}
D_1={2, 3, 4, 5}
2. z = ∞
1
2 3 4 5
2 3 4 5
7. z = 23
2 3
1 2 3
1 2
D_2={2}
D_2={3, 4, 5}
D_1={2}
D_1={3, 4, 5}
3. z = ∞
6. z = 25
8. z = 23
9. z = 22
1
2
3 4 5
1
3
2
2
3
1
3
1
D_3={3}
23
22
infeasible
4. z = ∞
5. z = 25
1
2
3
1
2
25
infeasible

Solve as an integer programming problem

$$\begin{aligned}
\text{min} \quad & 4x_1 + 3x_2 + 5x_3 \\
\text{subject to} \quad & 4x_1 + 2x_2 + 4x_3 \geq 17 \\
& x_j \leq (x_k - 1) + 5(1 - y_{jk}), \quad \text{all } j,k \text{ with } j < k \\
& x_k \leq (x_j - 1) + 5y_{jk}, \quad \text{all } j,k \text{ with } j < k \\
& 1 \leq x_j \leq 5, \quad x_j \text{ integer}, \quad j = 1,2,3 \\
& y_{jk} \in \{0,1\}, \quad \text{all } j,k \text{ with } j < k
\end{aligned}$$

Big-M constraints

$x_j < x_k$ if $y_{jk} = 1$

Linear relaxation

Use a linear programming algorithm to solve a continuous relaxation of the problem at each node of the search tree to obtain a lower bound on the optimal value of the problem at that node.

$$
\begin{aligned}
\text{min} \quad & 4x_1 + 3x_2 + 5x_3 \\
\text{subject to} \quad & 4x_1 + 2x_2 + 4x_3 \geq 17 \\
& x_j \leq (x_k - 1) + 5(1 - y_{jk}), \quad \text{all } j,k \text{ with } j < k \\
& x_k \leq (x_j - 1) + 5y_{jk}, \quad \text{all } j,k \text{ with } j < k \\
& 1 \leq x_j \leq 5, \quad j = 1,2,3 \\
& 0 \leq y_{jk} \leq 1, \quad \text{all } j,k \text{ with } j < k
\end{aligned}
$$

} Relax integrality

Alternate model

The following model has a better relaxation and would be used for this problem in practice. The big-*M* construction is used here to illustrate a popular and general technique.

$$
\begin{aligned}
\text{min} \quad & 4x_1 + 3x_2 + 5x_3 \\
\text{subject to} \quad & 4x_1 + 2x_2 + 4x_3 \geq 17 \\
& x_i = \sum_{j=1}^{5} j y_{ij}, \quad i = 1,2,3 \\
& \sum_{j=1}^{5} y_{ij} = 1, \quad i = 1,2,3 \\
& y_{jk} \in \{0,1\}, \quad \text{all } j,k
\end{aligned}
$$

Cutting planes

Infer the cutting planes

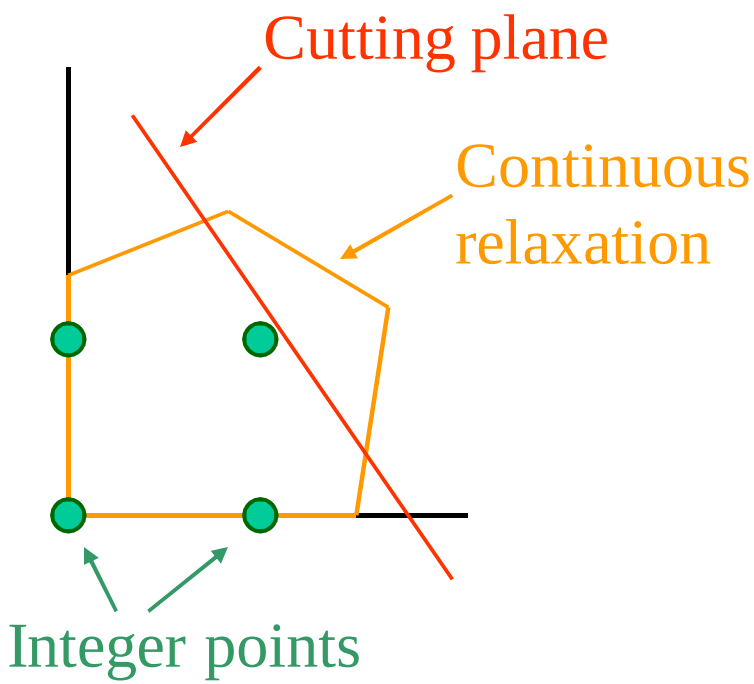
$$x_1 + x_2 + x_3 \geq 5$$

$$2x_1 + x_2 + 2x_3 \geq 9$$

From the inequality

$$4x_1 + 2x_2 + 4x_3 \geq 17$$

The cutting plane is implied by the inequality but strengthens the continuous relaxation



(One could also use the all-different constraint to obtain the stronger cutting plane $x_1 + x_2 + x_3 \geq 6$)

Branch and bound

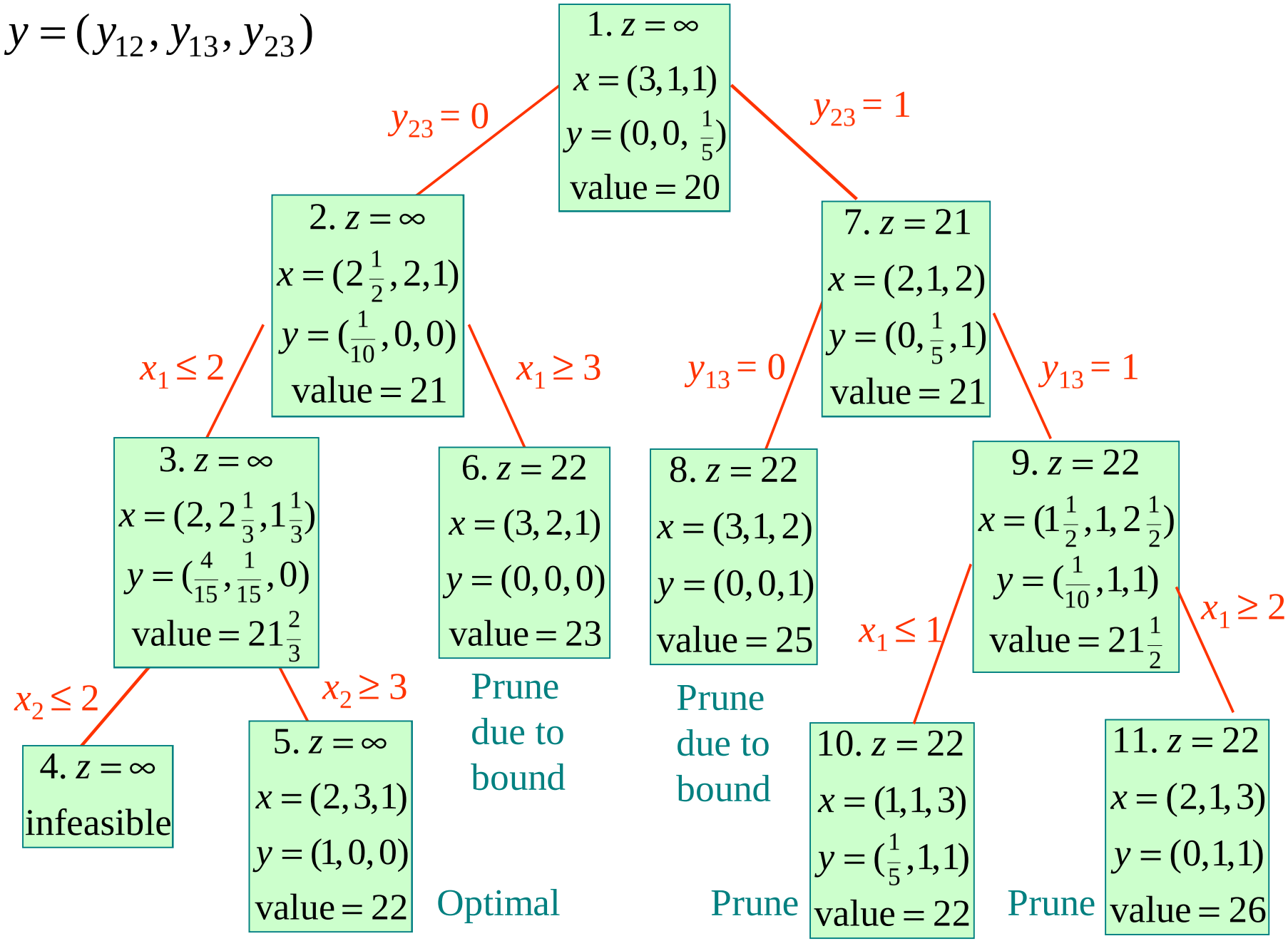
The *incumbent solution* is the best feasible solution found so far.

At each node of the branching tree:

- If Optimal value of relaxation $\geq$ Value of incumbent solution

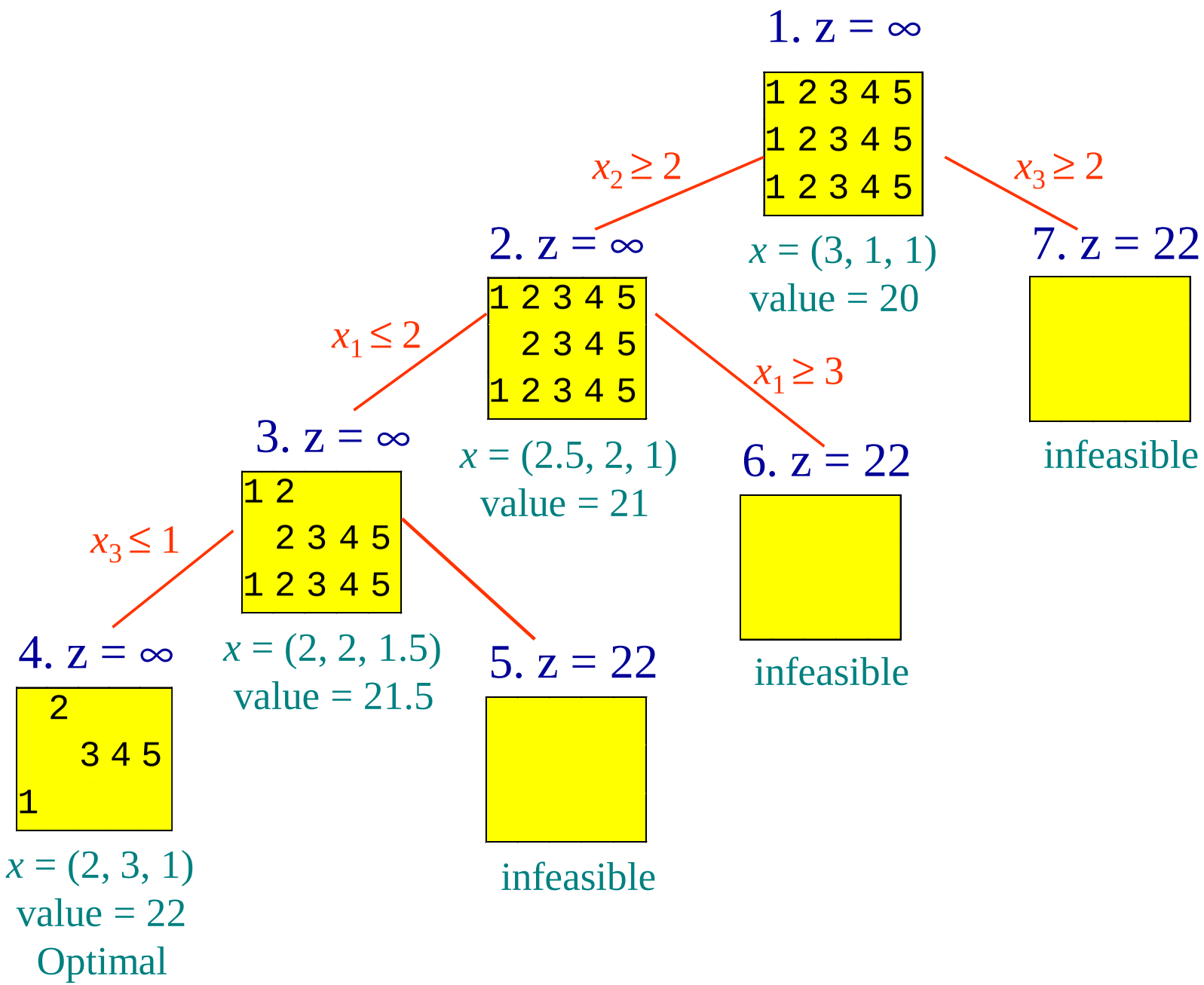
There is no need to branch further.

- No feasible solution in that subtree can be better than the incumbent solution.


$y = (y_{12}, y_{13}, y_{23})$
1. $z = \infty$
$x = (3,1,1)$
$y = (0,0,\frac{1}{5})$
value = 20
$y_{23} = 0$
$y_{23} = 1$
2. $z = \infty$
$x = (2\frac{1}{2},2,1)$
$y = (\frac{1}{10},0,0)$
value = 21
7. $z = 21$
$x = (2,1,2)$
$y = (0,\frac{1}{5},1)$
value = 21
$x_1 \le 2$
$x_1 \ge 3$
$y_{13} = 0$
$y_{13} = 1$
3. $z = \infty$
$x = (2,2\frac{1}{3},1\frac{1}{3})$
$y = (\frac{4}{15},\frac{1}{15},0)$
value = $21\frac{2}{3}$
6. $z = 22$
$x = (3,2,1)$
$y = (0,0,0)$
value = 23
8. $z = 22$
$x = (3,1,2)$
$y = (0,0,1)$
value = 25
9. $z = 22$
$x = (1\frac{1}{2},1,2\frac{1}{2})$
$y = (\frac{1}{10},1,1)$
value = $21\frac{1}{2}$
$x_2 \le 2$
$x_2 \ge 3$
$x_1 \le 1$
$x_1 \ge 2$
4. $z = \infty$
infeasible
5. $z = \infty$
$x = (2,3,1)$
$y = (1,0,0)$
value = 22
Prune due to bound
Prune due to bound
10. $z = 22$
$x = (1,1,3)$
$y = (\frac{1}{5},1,1)$
value = 22
11. $z = 22$
$x = (2,1,3)$
$y = (0,1,1)$
value = 26
Optimal
Prune
Prune

Combined approach

- Retain the *useful part* of the continuous relaxation - original inequalities plus cutting planes.
  - Omit big-M constraints, which add overhead to the relaxation while not improving the bound much.
  - Because relaxation is distinguished from model, both are succinct.
- Use bounds propagation on cutting planes as well as original inequality constraints.
- Maintain hyperarc consistency for all-different.
- Branch on nonintegral variable when possible; otherwise branch by splitting domain.


1. z = ∞
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
x = (3, 1, 1)
value = 20
x2 ≥ 2
x3 ≥ 2
7. z = 22
infeasible
2. z = ∞
1 2 3 4 5
2 3 4 5
1 2 3 4 5
x = (2.5, 2, 1)
value = 21
x1 ≤ 2
x1 ≥ 3
6. z = 22
infeasible
3. z = ∞
1 2
2 3 4 5
1 2 3 4 5
x = (2, 2, 1.5)
value = 21.5
x3 ≤ 1
5. z = 22
infeasible
4. z = ∞
2
3 4 5
1
x = (2, 3, 1)
value = 22
Optimal

Combined approach

Inference and Structure

- Search looks for a certificate of feasibility; i.e., a *solution*.
- Inference looks for a certificate of infeasibility (or optimality); i.e., a *proof*.
- Certificates of feasibility generally have polynomial length (i.e., most problems belong to *NP).*
- Certificates of infeasibility generally have exponential length (i.e., most problems don't belong to co-*NP).*
- The search for a proof therefore tends to bog down more readily than a search for a solution. Successful inference relies heavily on the identification of special structure.

The search/inference duality

- Two interpretations:
  - Complementary solution methods that can work together.
  - A formal mathematical duality that can lead to new methods.

The search/inference duality

- Complementary solution methods:
  - Search alone may find a good solution early, but it must examine many other solutions to determine that it is good.
  - Inference can rule out families of inferior solutions, but this is not the same as finding a good solution.
  - Working together, search & inference can find and verify good solutions more quickly.

The search/inference duality

- A formal duality:
  - Search and inference are related by a formal optimization duality
  - Linear programming duality is a special case.
  - This provides a general method for sensitivity analysis.
  - It also provides a general form of Benders decomposition, which is closely related to the use of nogoods.

Outline

- A motivating example
  - Constraint satisfaction approach
  - Integer programming approach
  - Combined approach
- The search/inference duality
  - Complementary solution methods
  - A formal duality
- The strengthening/relaxation duality
  - Complementary solution methods
  - Relaxations for global constraints
  - Formal relaxation duality
- Relaxation duality
  - An integer programming example
  - Continuous relaxation
  - Discrete relaxation
  - *dependency graph, nonserial dynamic programming*
  - Relaxation duality
  - Lagrangean & surrogate duals
  - Discrete relax + Lagrangean dual
  - Discrete relaxation dual
  - Discrete + Lagrangean duals
  - Summary of relaxations
- Research agenda

The strengthening/relaxation duality

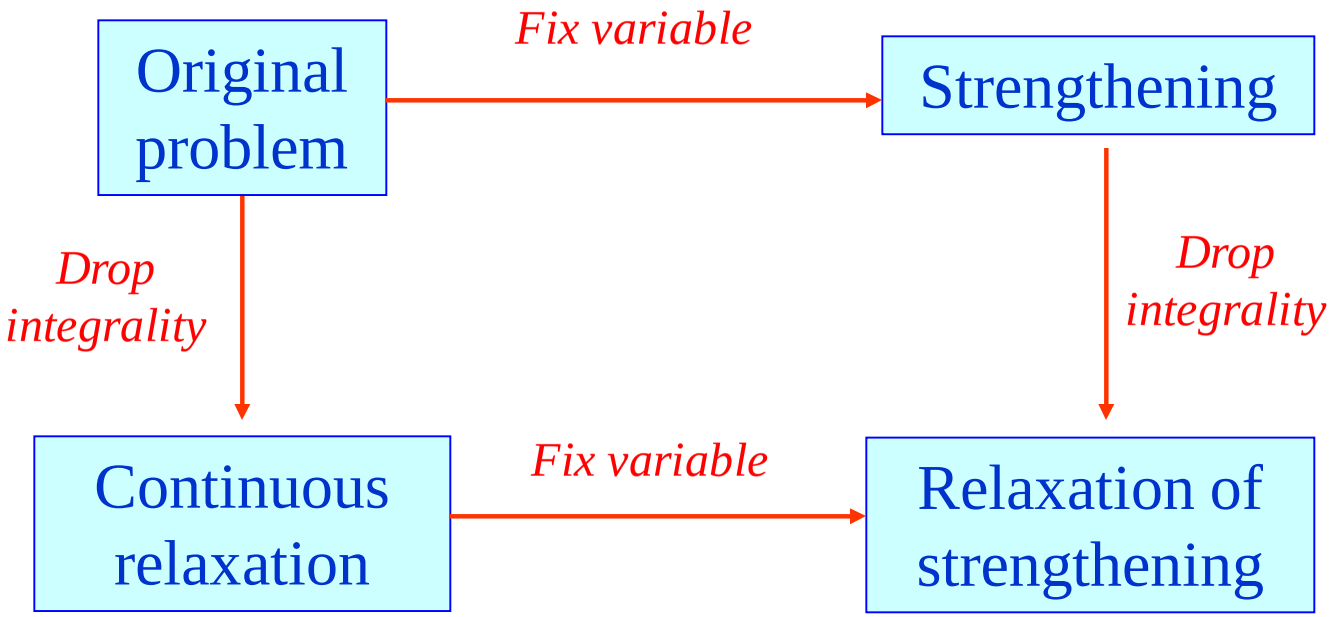
- Three interpretations:
  - Complementary solution methods that can work together.
  - Creation of relaxations as well as domain reduction algorithms to exploit structure of subsets of constraints (e.g., element constraints).
  - A formal mathematical duality that can lead to new relaxations, particularly for constraint satisfaction models.

The strengthening/relaxation duality

- Complementary solution methods
  - Branch-and-bound solves relaxations of strengthenings. Branching creates strengthenings, and one solves a relaxation of each to obtain bounds.
  - There are other ways strengthening and relaxation can relate. One can solve strengthenings of a relaxation. Branching creates strengthenings of an initial relaxation.

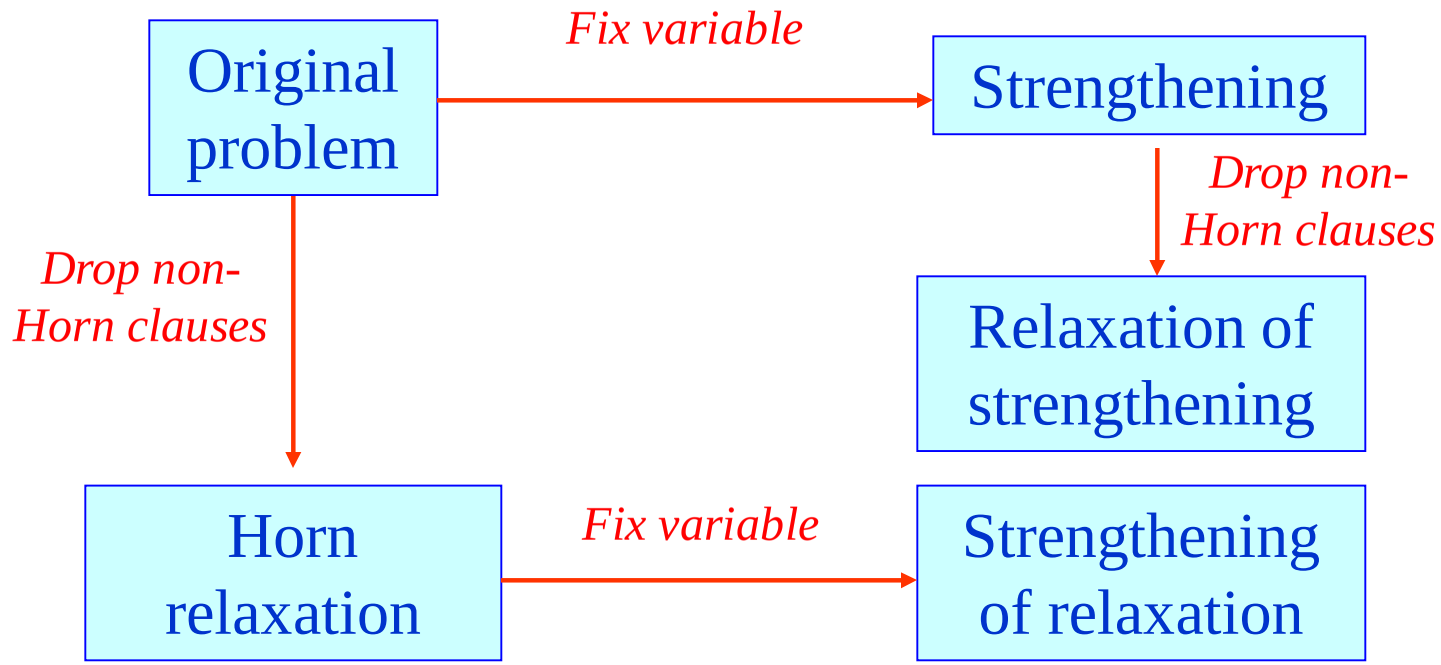
Relaxations of strengthenings vs. strengthenings of a relaxation

They are the same in integer programming because the strengthening and relaxation functions commute:

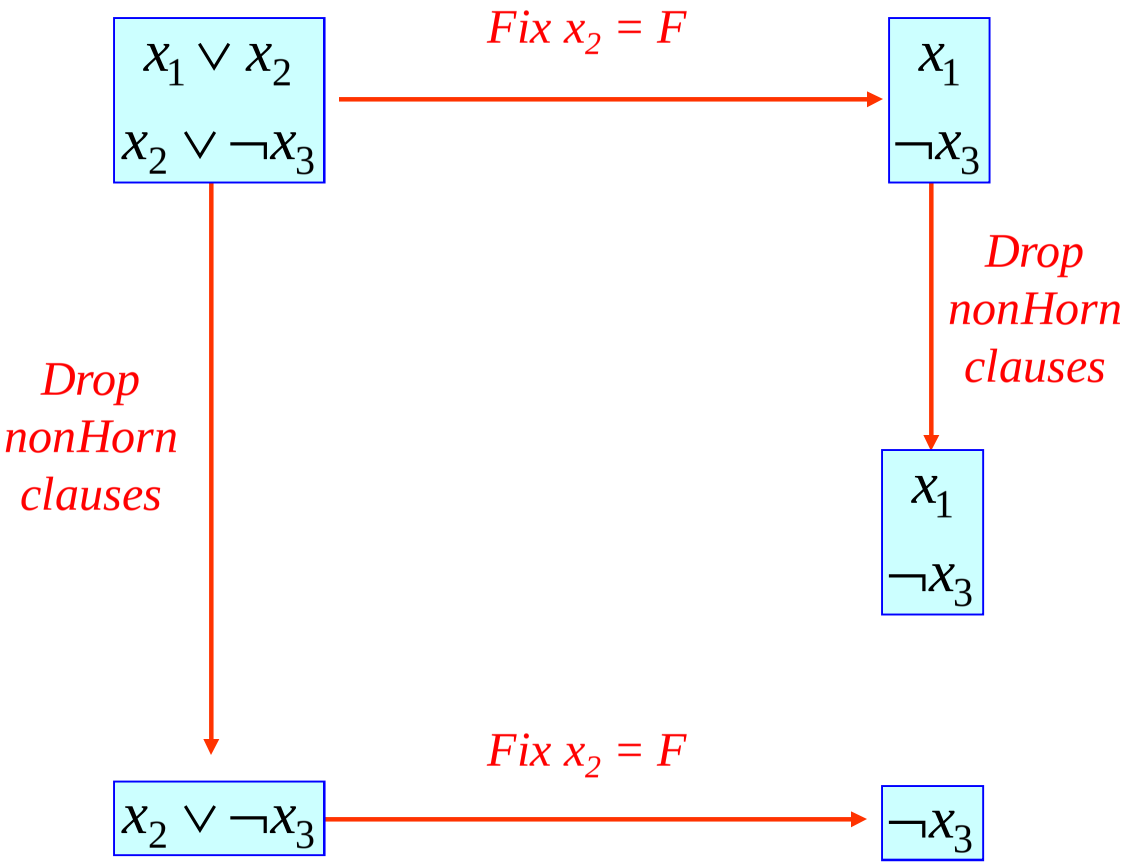


Relaxations of strengthenings vs. strengthenings of a relaxation

They are not the same in general; for example, when solving a propositional satisfiability problem with the help of Horn relaxation. Diagram does not commute.



Example...


$x_1 \vee x_2$
$x_2 \vee \neg x_3$
Fix x_2 = *F*
x_1
$\neg x_3$
Drop nonHorn clauses
Drop nonHorn clauses
x_1
$\neg x_3$
$x_2 \vee \neg x_3$
Fix x_2 = *F*
$\neg x_3$

The strengthening/relaxation duality

- Another interpretation: create relaxations as well as domain reduction algorithms for specially structured "global" constraints.
  - This will be illustrated with the element constraint, which can be used to implement variable subscripts (indices).

Discrete variable with variable index

The constraint $x_y \leq \beta$ where $x_j \in D_{xj}$, $y \in D_y$ are discrete variables, can be implemented: $z \leq \beta$

$$\text{element}(y, (x_1, \ldots, x_n), z)$$

Here, element is processed with a discrete domain reduction algorithm that maintains hvperarc consistency.

$$D_z \leftarrow D_z \cap \bigcup_{j \in D_y} D_{x_j}$$

$$D_y \leftarrow D_y \cap \{ j \mid D_x \cap D_{x_j} \neq \emptyset \}$$

$$D_{x_j} \leftarrow \begin{cases} D_z & \text{if } D_y = \{ j \} \\ D_{x_j} & \text{otherwise} \end{cases}$$

Example... element$(y,(x_1,x_2,x_3,x_4),z)$

The initial domains are:

$D_z = \{20,30,60,80,90\}$

$D_y = \{1,3,4\}$

$D_{x_1} = \{10,50\}$

$D_{x_2} = \{10,20\}$

$D_{x_3} = \{40,50,80,90\}$

$D_{x_4} = \{40,50,70\}$

The reduced domains are:

$D_z = \{80,90\}$

$D_y = \{3\}$

$D_{x_1} = \{10,50\}$

$D_{x_2} = \{10,20\}$

$D_{x_3} = \{80,90\}$

$D_{x_4} = \{40,50,70\}$

Continuous variable with variable index

The constraint $x_y \leq \beta$ where each x_j $(0 \leq x_j \leq m_j)$ is a continuous variable, can be implemented: $z \leq \beta$

$$\text{element}(y, (x_1, \ldots, x_n), z)$$

Here, element generates a continuous relaxation that is added to the linear programming subproblem:

$$\sum_{i \in D_y} \frac{x_i}{m_i} - \left(\sum_{i \in D_y} \frac{1}{m_i} \right) z \geq -|D_y| + 1$$

$$-\sum_{i \in D_y} \frac{x_i}{m_i} + \left(\sum_{i \in D_y} \frac{1}{m_i} \right) z \geq -|D_y| + 1$$

Example...

$$\text{element}(y, (x_1, x_2), z)$$

$$0 \le x_1 \le 4$$

$$0 \le x_2 \le 5$$

The relaxation is:

$$5x_1 + 4x_2 - 9z \le 20$$

$$5x_1 + 4x_2 - 9z \ge -20$$

$$0 \le x_1 \le 4$$

$$0 \le x_2 \le 4$$

The strengthening/relaxation duality

- Can be interpreted as a formal relaxation duality.
  - Linear programming duality, Lagrangean duality, surrogate duality are special cases.
  - These classical dualities apply only to numeric equality and inequality constraints.
  - General relaxation duality can be used to create new relaxations for other constraints.
  - One approach is to use the concept of induced width of a dependency graph, along with nonserial dynamic programming.

Outline

- A motivating example
  - Constraint satisfaction approach
  - Integer programming approach
  - Combined approach
- The search/inference duality
  - Complementary solution methods
  - A formal duality
- The strengthening/relaxation duality
  - Complementary solution methods
  - Relaxations for global constraints
  - Formal relaxation duality
- Relaxation duality
  - An integer programming example
  - Continuous relaxation
  - Discrete relaxation
  - *dependency graph, nonserial dynamic programming*
  - Relaxation duality
  - Lagrangean & surrogate duals
  - Discrete relax + Lagrangean dual
  - Discrete relaxation dual
  - Discrete + Lagrangean duals
  - Summary of relaxations
- Research agenda

Integer programming example

$$
\begin{aligned}
\text{min} \quad & 35x_1 + 20x_2 + 15x_3 + 40x_4 + 30x_5 \\
\text{subject to} \quad & 2x_1 + 3x_2 + x_3 \geq 4 \\
& x_2 + 4x_3 + 3x_4 \geq 3 \\
& 2x_3 + 3x_4 + x_5 \geq 4 \\
& 3x_1 + 4x_3 + 5x_5 \geq 5 \\
& x_j \in \{0,1\}, \quad \text{all } j
\end{aligned}
$$

Optimal value = 105

Continuous relaxation

$$
\begin{aligned}
\text{min} \quad & 35x_1 + 20x_2 + 15x_3 + 40x_4 + 30x_5 \\
\text{subject to} \quad & 2x_1 + 3x_2 + x_3 \geq 4 \\
& x_2 + 4x_3 + 3x_4 \geq 3 \\
& 2x_3 + 3x_4 + x_5 \geq 4 \\
& 3x_1 + 4x_3 + 5x_5 \geq 5 \\
& 0 \leq x_j \leq 1, \quad \text{all } j
\end{aligned}
$$

Optimal value = $43\frac{1}{3}$

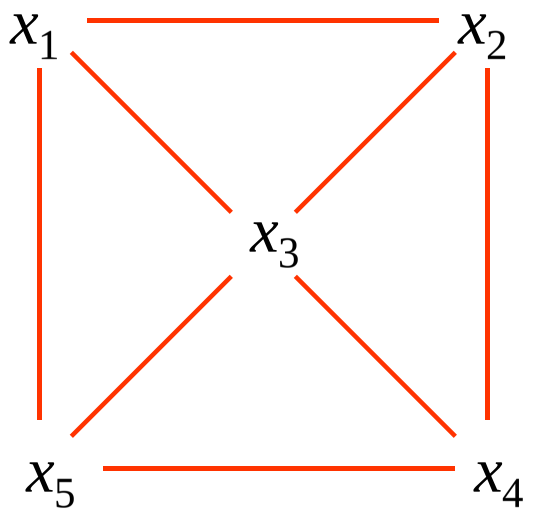
Dependency graph

$$2x_1 + 3x_2 - x_3 \geq 4$$

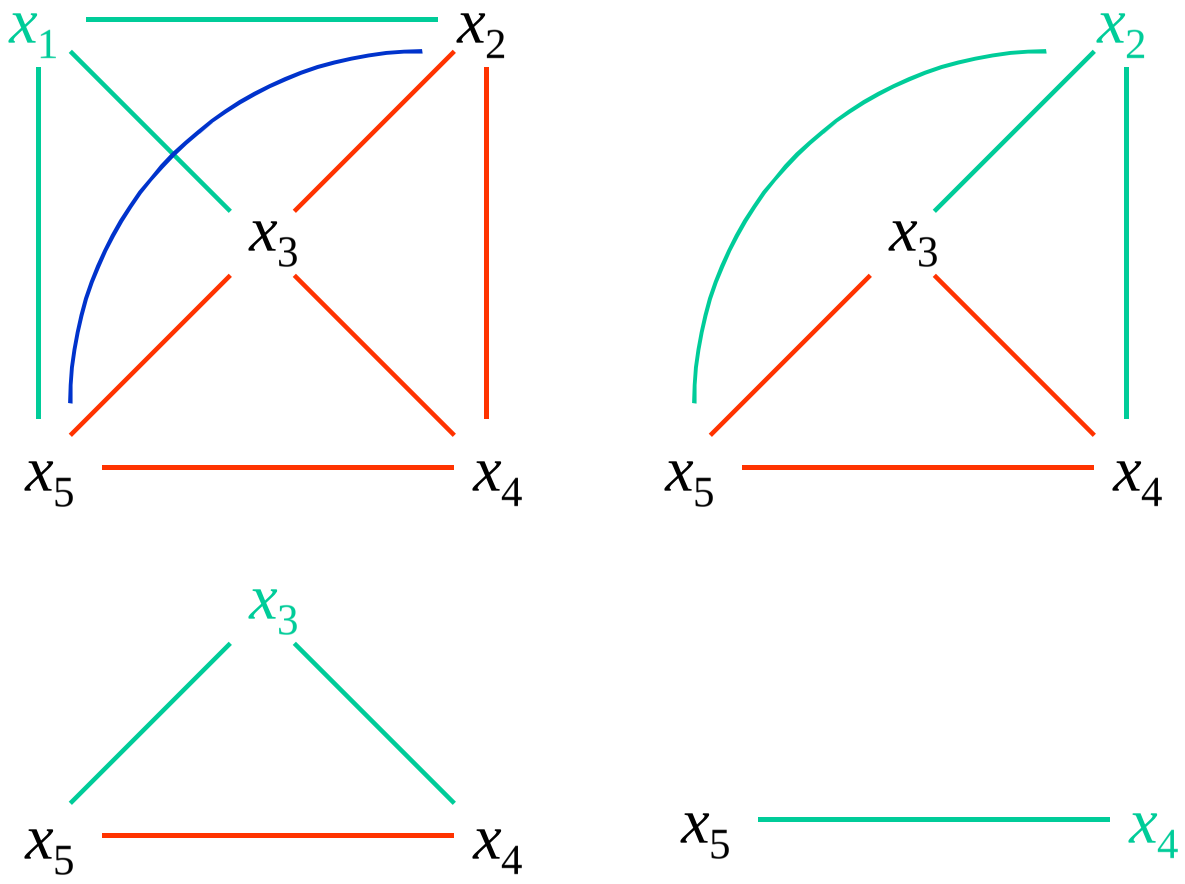
$$x_2 + 4x_3 + 3x_4 \geq 3$$

$$2x_3 + 3x_4 + x_5 \geq 4$$

$$3x_1 + 4x_3 + 5x_5 \geq 5$$



Induced width = 3

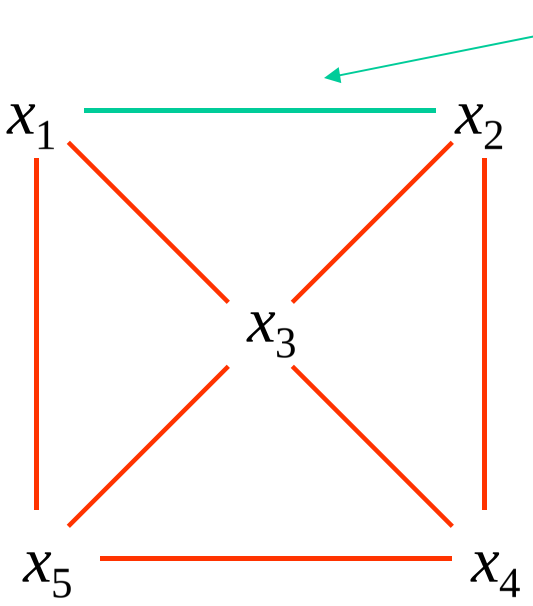


Discrete relaxation

To create a discrete relaxation:

- Thin out the dependency graph so that it has a smaller induced width.
- Use projection to remove variable couplings that correspond to deleted arcs.
- Solve resulting problem by nonserial dynamic programming, whose complexity varies exponentially with induced width.
    - The idea of nonserial dynamic programming has surfaced in several contexts: Markov trees, solution of Bayesian networks, etc.

Reduce induced width to 2



$2x_1 + 3x_2 + x_3 \geq 4$

Project onto x_2, x_3

$3x_2 + x_3 \geq 1$

Project onto x_1, x_3

$2x_1 + x_3 \geq 1$

This removes coupling between x_1, x_2 in constraint 1

Discrete relaxation

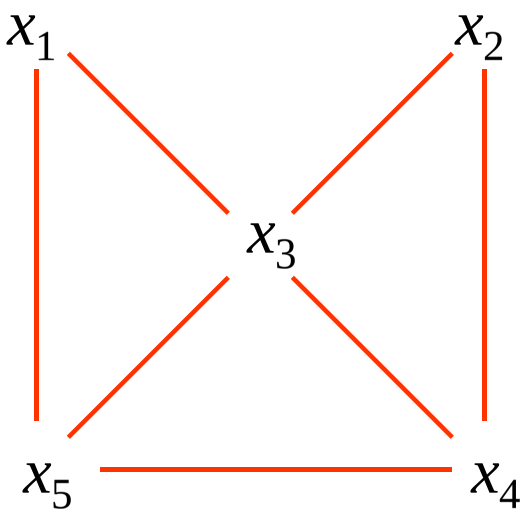
$$
\begin{aligned}
\text{min} \quad & 35x_1 + 20x_2 + 15x_3 + 40x_4 + 30x_5 \\
\text{subject to} \quad & 3x_2 + x_3 \geq 2 \\
& 2x_1 + x_3 \geq 1 \\
& x_2 + 4x_3 + 3x_4 \geq 3 \\
& 2x_3 + 3x_4 + x_5 \geq 4 \\
& 3x_1 + 4x_3 + 5x_5 \geq 5 \\
& x_j \in \{0,1\}, \quad \text{all } j
\end{aligned}
$$

Optimal value = 105 (same as original problem)

Nonserial dynamic programming

$$g_1(x_3, x_5) = \min_{x_1}\{35x_1 + 15x_3 + 30x_5 + M(1 - 2x_1 - x_3)^+ + M(5 - 3x_1 - 4x_3 - 5x_5)^+\}$$

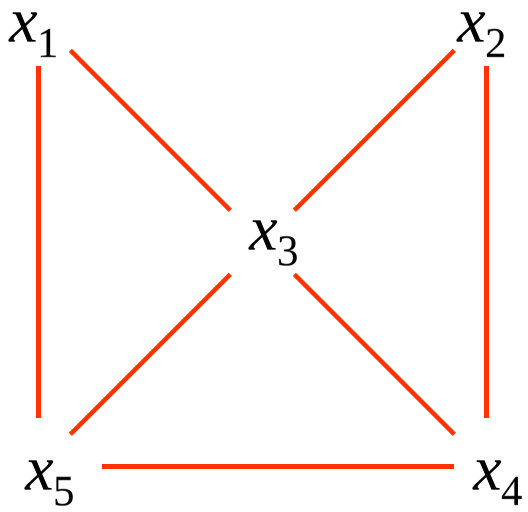
$$g_2(x_3, x_4) = \min_{x_2}\{20x_2 + 40x_4 + M(2 - 3x_2 - x_3)^+ + M(3 - x_2 - 4x_3 - 3x_4)^+\}$$



NSDP, continued

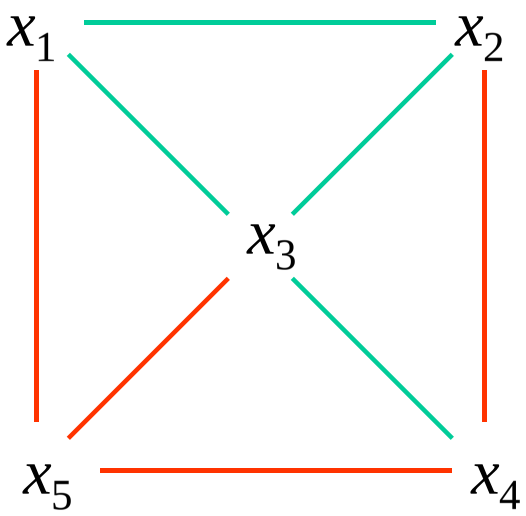
$$g_3(x_4, x_5) = \min_{x_3}\{g_1(x_3, x_5) + g_2(x_3, x_4) + M(4 - 2x_3 - 3x_4 - x_5)^+\}$$

$$\text{solution} = \min_{x_4, x_5} g_3(x_4, x_5) = 105$$



Reduce induced width to 1

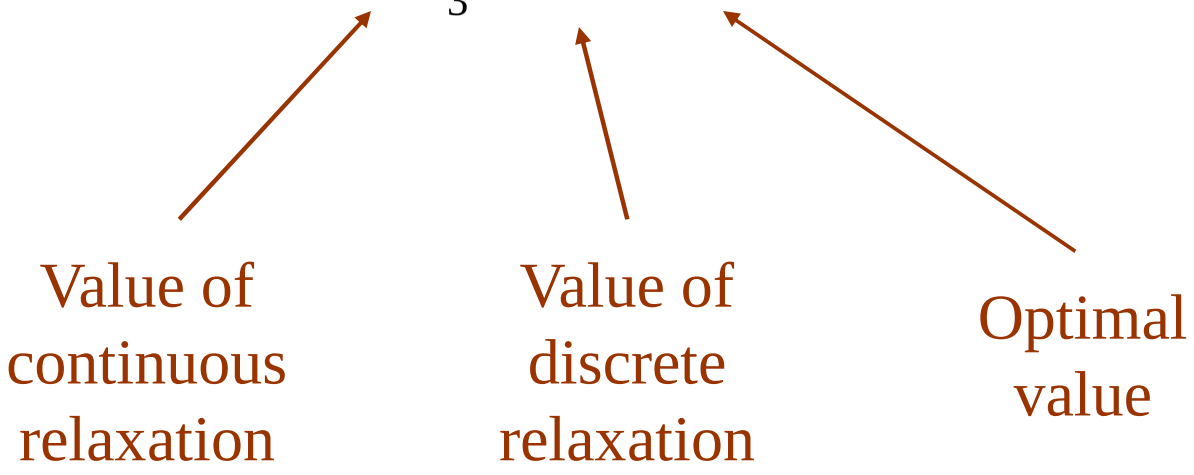
$$\begin{aligned}
&\min && 35x_1 + 20x_2 + 15x_3 + 40x_4 + 30x_5 \\
&\text{subject to} && 3x_2 \geq 1 \\
& && 3x_4 + x_5 \geq 2 \\
& && 2x_3 + x_5 \geq 1 \\
& && 4x_3 + 5x_5 \geq 2 \\
& && 3x_1 + 5x_5 \geq 1 \\
& && x_j \in \{0,1\}, \quad \text{all } j
\end{aligned}$$



Optimal value = 90

Continuous & discrete relaxations

$$43\tfrac{1}{3} < 90 < 105$$



Parameterized relaxation

$$\begin{aligned} &\min && f(x) \\ &\text{subject to} && x \in S \end{aligned}$$ } Generic optimization problem

$$\begin{aligned} \theta(\lambda) = \quad &\min && f(x,\lambda) \\ &\text{subject to} && x \in S(\lambda) \end{aligned}$$ } Parameterized relaxation

$$\max_{\lambda \in \Lambda} \{\theta(\lambda)\}$$ } Relaxation dual

General conditions for a relaxation:

$$f(x,\lambda) \leq f(x), \quad \text{all } x \in S, \lambda \in \Lambda$$

$$S(\lambda) \supset S, \quad \text{all } \lambda \in \Lambda$$

Lagrangean relaxation

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & g_i(x) \le 0, \quad i \in I \\ & x \in S \end{aligned}$$

} Optimization problem

$$\theta(\lambda) = \min \; f(x) + \sum_{i \in I} g_i(x) \quad \text{subject to} \quad x \in S$$

} Lagrangean relaxation

$$\max_{\lambda \ge 0} \{\theta(\lambda)\}$$

} Lagrangean dual

Surrogate dual

$$\begin{aligned} &\min && f(x) \\ &\text{subject to} && g_i(x) \le 0, \quad i \in I \\ & && x \in S \end{aligned}$$

} Optimization problem

$$\begin{aligned} \theta(\lambda) = \quad &\min && f(x) \\ &\text{subject to} && \sum_{i \in I} \lambda_i g_i(x) \le 0 \\ & && x \in S \end{aligned}$$

} Surrogate relaxation

$$\max_{\lambda \ge 0} \{\theta(\lambda)\}$$

} Surrogate dual

Combine discrete relaxation with Lagrangean duality

$$
\begin{aligned}
\min \quad & 35x_1 + 20x_2 + 15x_3 + 40x_4 + 30x_5 \\
& + \lambda_1(4 - 2x_1 - 3x_2 - x_3) \\
& + \lambda_2(3 - x_2 - 4x_3 - 3x_4) \\
& + \lambda_3(4 - 2x_3 - 3x_4 - x_5) \\
& + \lambda_4(5 - 3x_1 - 4x_3 - 5x_5)
\end{aligned}
$$

} Lagrangean

$$
\begin{aligned}
\text{subject to} \quad & 3x_2 \geq 1 \\
& 3x_4 + x_5 \geq 2 \\
& 2x_3 + x_5 \geq 1 \\
& 4x_3 + 5x_5 \geq 2 \\
& 3x_1 + 5x_5 \geq 1 \\
& x_j \in \{0,1\}, \quad \text{all } j
\end{aligned}
$$

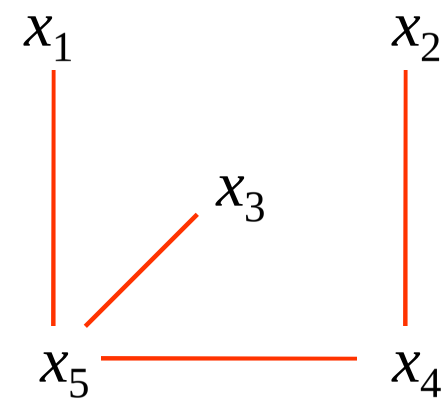
} Discrete relaxation

Solve by subgradient optimization

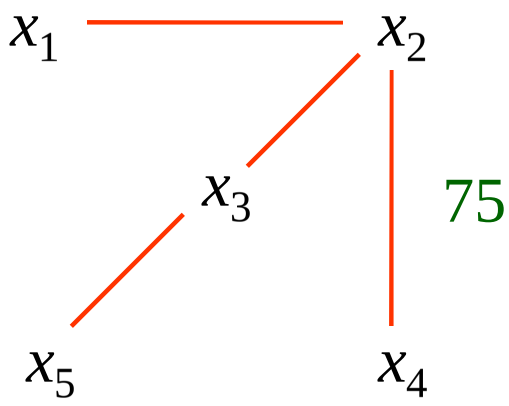
Step	λ	Value	Subgradient
	0 0 0 0	90	1 -1 -4 0
6	6 0 0 0	96	1 -1 -4 0
3	9 0 0 0	92	-2 -5 -1 0
2	5 0 0 2	95	1 -1 -4 0
1.5	6.5 0 0 2	96.5	1 -1 -4 0
1.2	7.7 0 0 2	96.6	-2 -5 -1 0
1	5.7 0 0 3	95.7	1 -1 -4 0
.86	6.56 0 0 3	96	0 -5 -6 -3
.75	6.56 0 0 .75	96.56	1 -1 -4 0
.67	7.23 0 0 .75	96.29	-2 -5 -1 1
.6	7.23 0 0 1.35	96.89	-2 -5 -1 1

Use NSDP at each iteration.

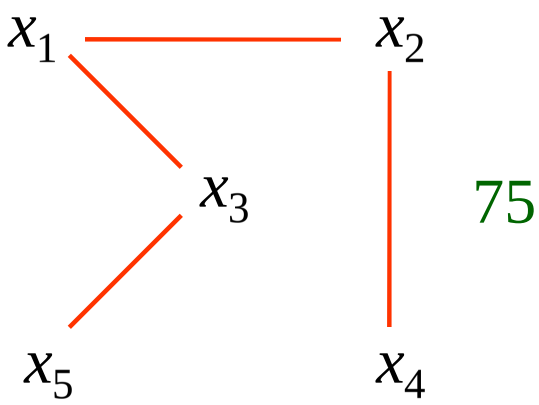
Discrete relaxation dual



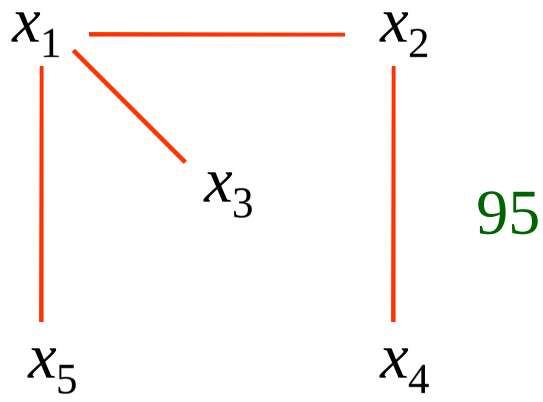
Bound
= 90



75



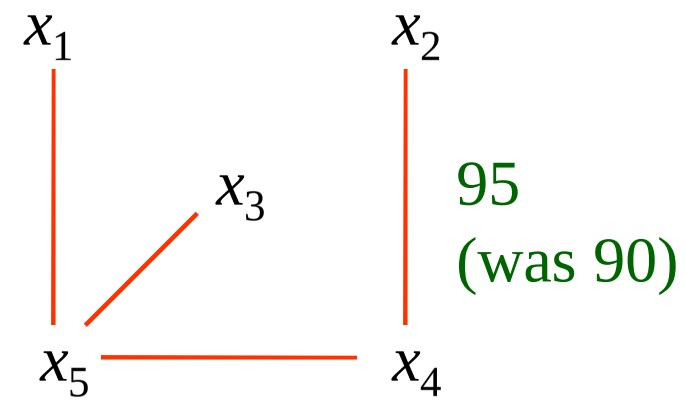
75



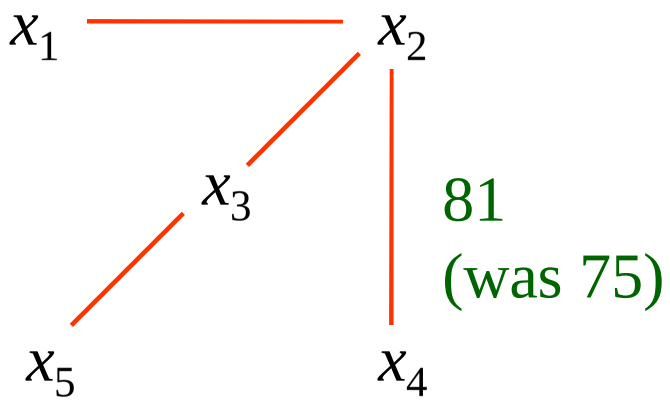
95

Enumerate relaxations with induced width of 1

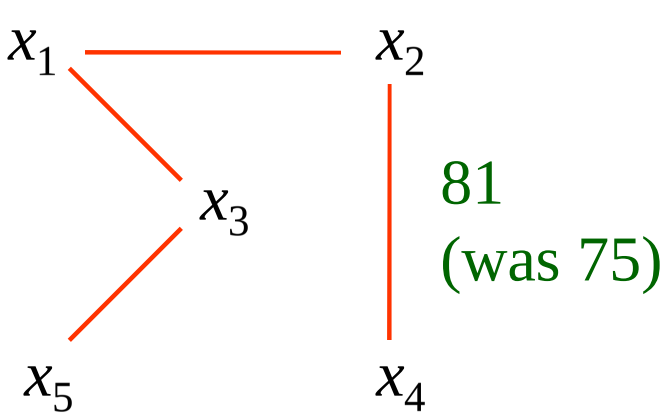
Combine Lagrangean & discrete relaxation duals



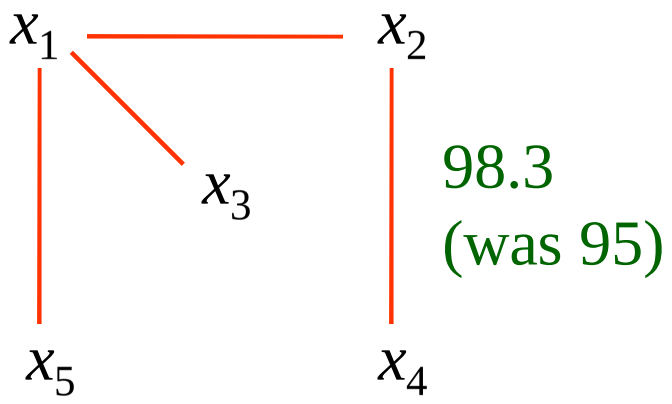
95
(was 90)



81
(was 75)

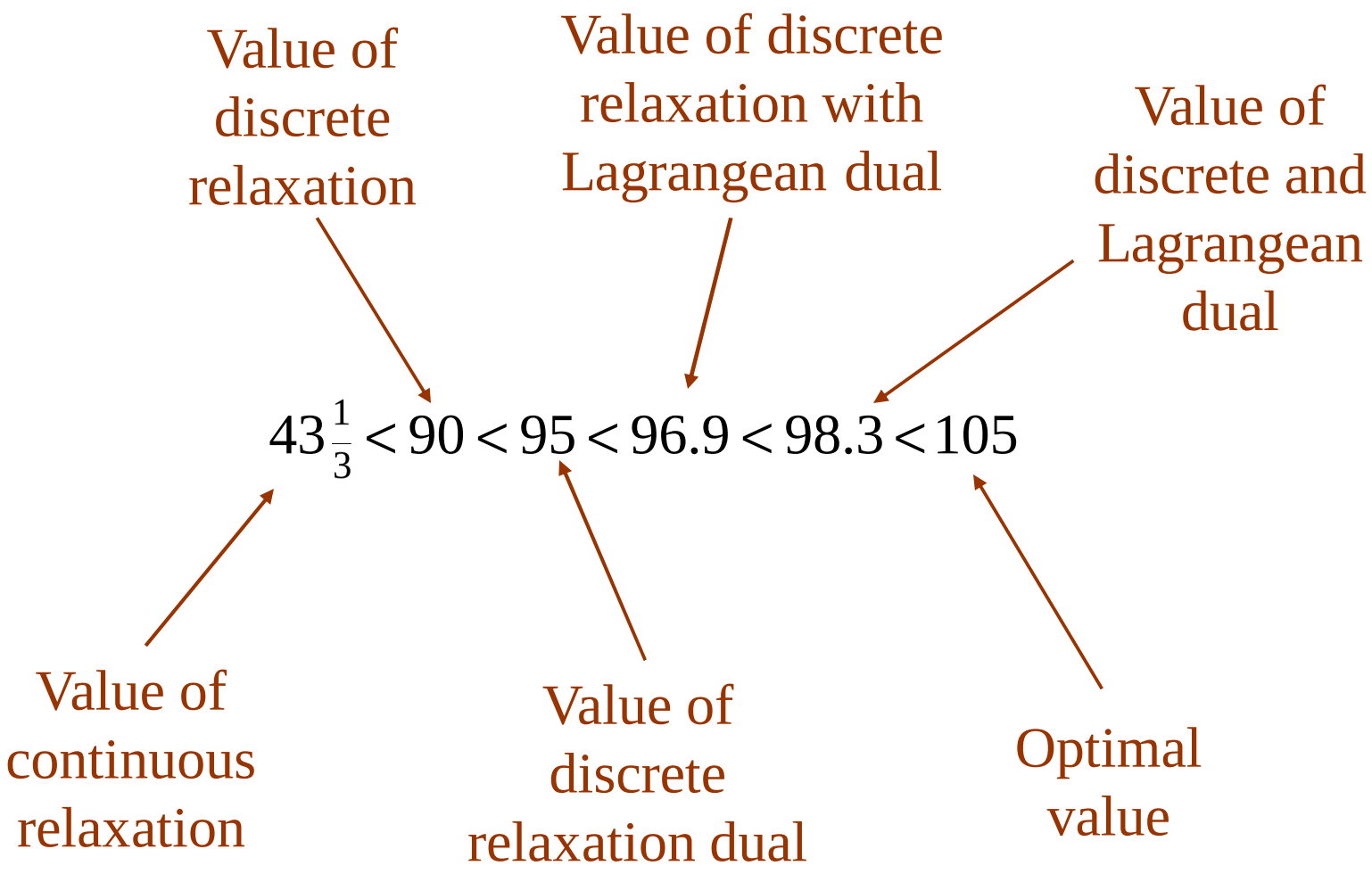


81
(was 75)



98.3
(was 95)

Summary of relaxations



Outline

- A motivating example
  - Constraint satisfaction approach
  - Integer programming approach
  - Combined approach
- The search/inference duality
  - Complementary solution methods
  - A formal duality
- The strengthening/relaxation duality
  - Complementary solution methods
  - Relaxations for global constraints
  - Formal relaxation duality
- Relaxation duality
  - An integer programming example
  - Continuous relaxation
  - Discrete relaxation
  - *dependency graph, nonserial dynamic programming*
  - Relaxation duality
  - Lagrangean & surrogate duals
  - Discrete relax + Lagrangean dual
  - Discrete relaxation dual
  - Discrete + Lagrangean duals
  - Summary of relaxations
- Research agenda

Research Agenda

- Identify cutting planes that propagate well.
- Learn how to choose constraints that have a useful continuous relaxation.
- Find continuous relaxations for global constraints not in inequality form (e.g., element, piecewise linear costs).
- Implement variable index sets as well as variable indices.
- Use relaxation duals to discover new relaxations common constraints in constraint satisfaction languages.

Research Agenda

- Identify inference techniques (other than cutting planes) that obtain relaxations that are easy to solve.
- Develop inference-based sensitivity analysis for problem classes.
- Investigate the possibility of using nogoods in branch-and-bound search along with cutting planes and domain reduction.
- Use generalized Benders decomposition to obtain useful nogoods.

Research Agenda

- Experiment with new ways to combine strengthening and relaxation.
- Solve a wide variety of problems with a view to how the search/inference and strengthening/relaxation dualities may be exploited.
- Build a solution technology that unifies and goes beyond classical optimization and constraint satisfaction methods.