Integer Optimization: **Mathematics, Algorithms, and Applications**

Sommerschool

Jacobs University, July 2007

DFG Research Center Matheon
Mathematics for key technologies

Thorsten Koch
Zuse Institute Berlin

Joint work with

Tobias Achterberg

Timo Berthold

Benjamin Hiller

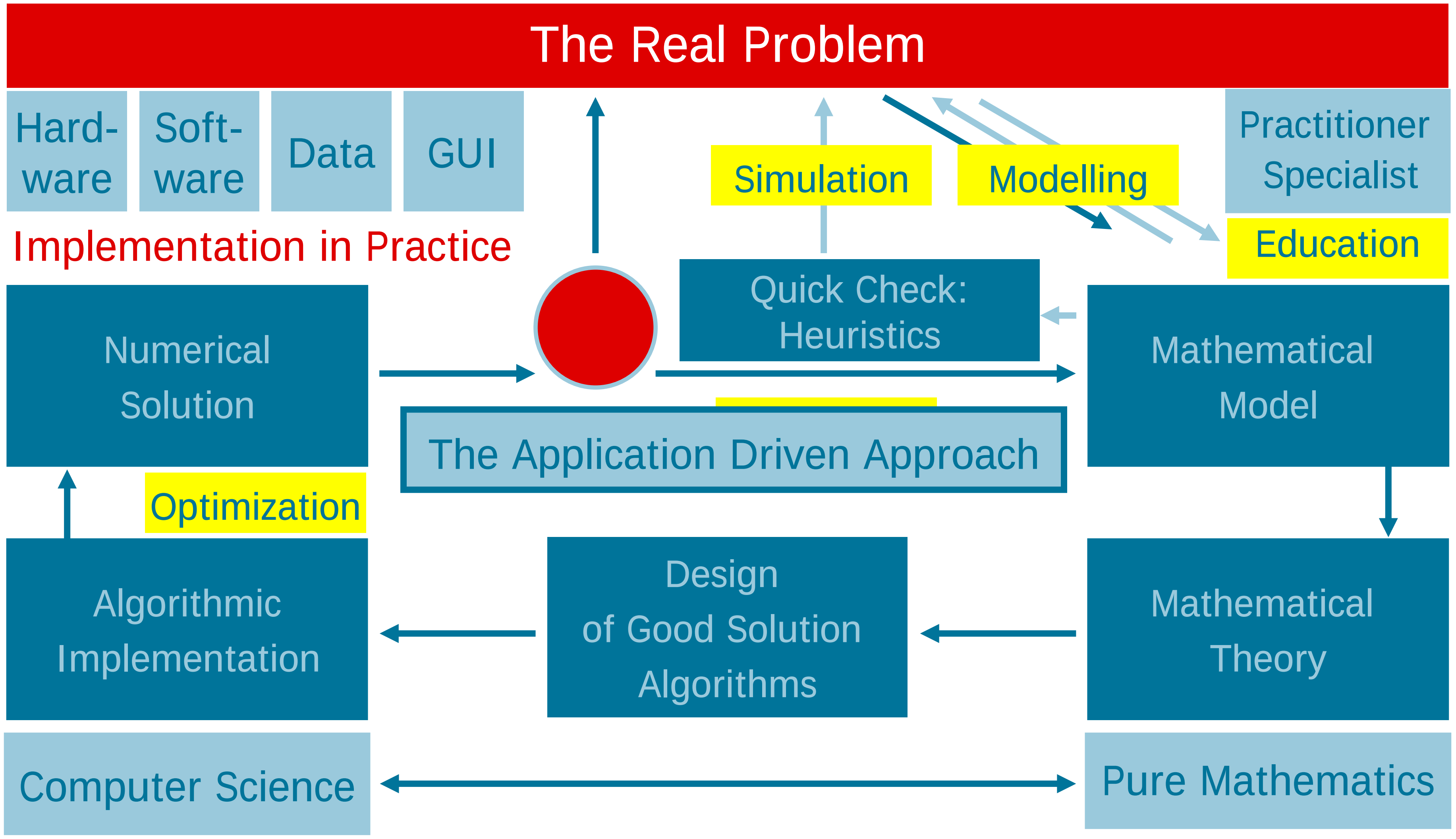
Sebastian Orlowski

Marc Pfetsch

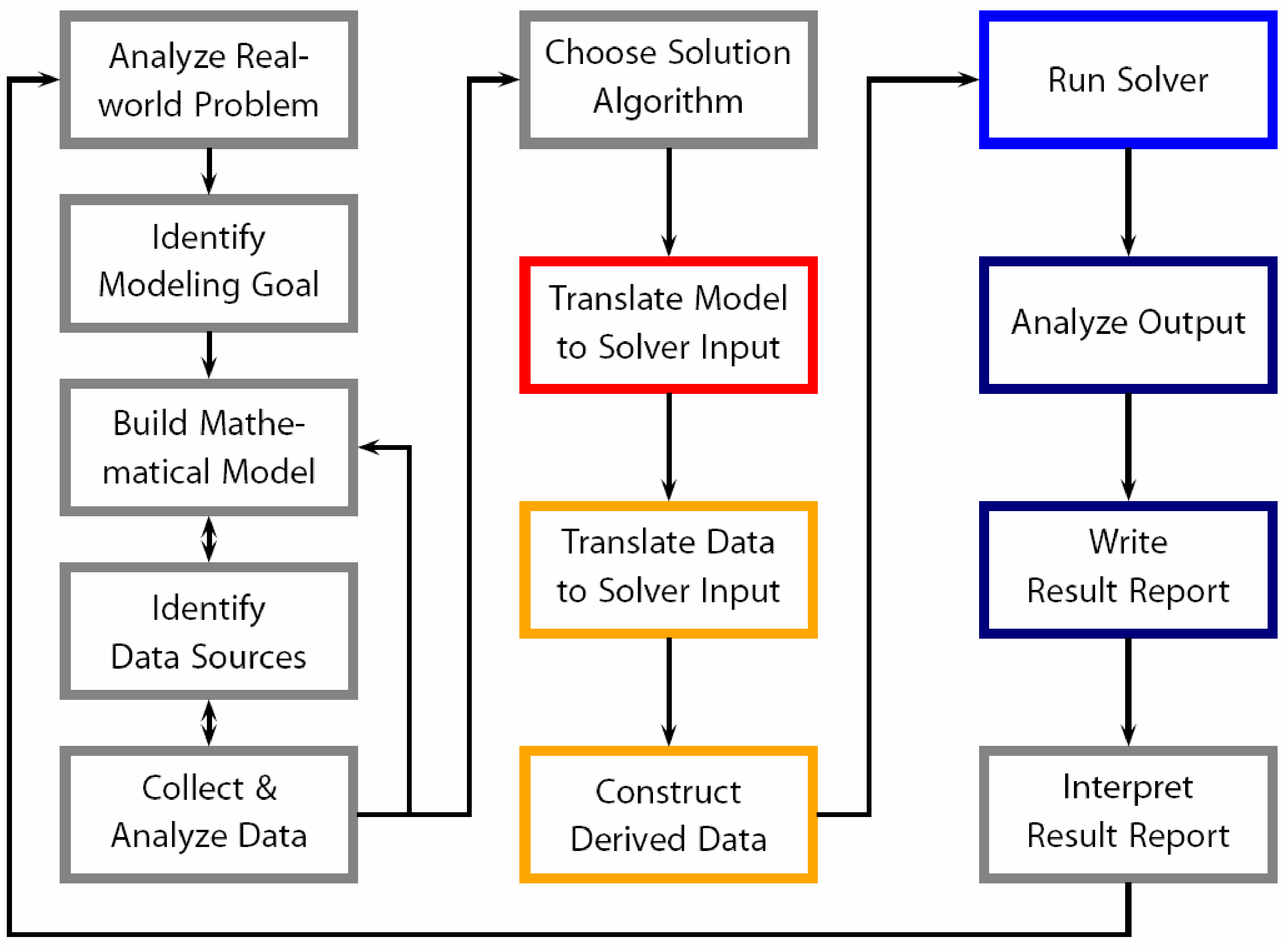
Andreas Tuchscherer

...

The Problem Solving Cycle in Modern Applied Mathematics



Modeling Cycle



typical optimization problems

$$\begin{aligned} &\max\ f(x)\ \ or\ \min\ f(x) \\ &g_i(x) = 0, \quad i = 1,2,\ldots,k \\ &h_j(x) \le 0, \quad j = 1,2,\ldots,m \\ &x \in \mathbf{R}^n\ (and\ \ x \in S) \end{aligned}$$

„general“
(nonlinear)
program
NLP

$$\begin{aligned} &\min c^T x \\ &Ax = a \\ &Bx \le b \\ &\quad x \ge 0 \\ &(x \in \mathbf{R}^{n\mathbf{n}}) \\ &(x \in \mathrm{k}^{n\mathbf{n}}) \end{aligned}$$

linear
program
LP

$$\begin{aligned} &\min c^T x \\ &Ax = a \\ &Bx \le b \\ &\quad x \ge 0 \\ &some\ x_j \in \mathbf{Z} \\ &(x \in \{0,1\}^n) \end{aligned}$$

(linear)
0/1-
mixed-
integer
program
IP, MIP

program = optimization problem

Linear Programming

$$\max c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

$$\textit{subject to}$$

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$$
$$\cdot$$
$$a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m$$
$$x_1, x_2, \ldots, x_n \geq 0$$

$$\max c^T x$$
$$Ax = b$$
$$x \geq 0$$

linear program
in standard form

Linear Programming

1939 L. V. Kantorovitch: Foundations of linear programming (Nobel Prize 1975)

1947 G. B. Dantzig: Invention of the simplex algorithm

$$\max c^T x$$
$$Ax = b$$
$$x \geq 0$$

Today: From an economic point of view,
linear programming is one of the most important developments of mathematics in the 20th century.

Stiglers „Diet Problem“:„The first linear program“

$\min x_1 + x_2$	costs
$2x_1 + x_2 \geq 3$	protein
$x_1 + 2x_2 \geq 3$	carbohydrates
$x_1 \geq 0$	potatoes
$x_2 \geq 0$	beans

minimizing the cost of food

George J. Stigler
Nobel Prize in economics 1982

Sets n nutrients / calorie thousands , protein grams , calcium grams , iron milligrams vitamin-a thousand ius, vitamin-b1 milligrams, vitamin-b2 milligrams, niacin milligrams , vitamin-c milligrams /

f foods / wheat , cornmeal , cannedmilk, margarine , cheese , peanut-b , lard liver , porkroast, salmon , greenbeans, cabbage , onions , potatoes spinach, sweet-pot, peaches , prunes , limabeans, navybeans /

Parameter b(n) required daily allowances of nutrients / calorie 3, protein 70 , calcium .8 , iron 12 vitamin-a 5, vitamin-b1 1.8, vitamin-b2 2.7, niacin 18, vitamin-c 75 /

Table a(f,n) nutritive value of foods (per dollar spent)

	calorie	protein	calcium	iron	vitamin-a	vitamin-b1	vitamin-b2	niacin	vitamin-c
	(1000)	(g)	(g)	(mg)	(1000iu)	(mg)	(mg)	(mg)	(mg)
wheat	44.7	1411	2.0	365		55.4	33.3	441	
cornmeal	36	897	1.7	99	30.9	17.4	7.9	106	
cannedmilk	8.4	422	15.1	9	26	3	23.5	11	60
margarine	20.6	17	.6	6	55.8	.2			
cheese	7.4	448	16.4	19	28.1	.8	10.3	4	
peanut-b	15.7	661	1	48		9.6	8.1	471	
lard	41.7				.2		.5	5	
liver	2.2	333	.2	139	169.2	6.4	50.8	316	525
porkroast	4.4	249	.3	37		18.2	3.6	79	
salmon	5.8	705	6.8	45	3.5	1	4.9	209	
greenbeans	2.4	138	3.7	80	69	4.3	5.8	37	862
cabbage	2.6	125	4	36	7.2	9	4.5	26	5369
onions	5.8	166	3.8	59	16.6	4.7	5.9	21	1184
potatoes	14.3	336	1.8	118	6.7	29.4	7.1	198	2522
spinach	1.1	106		138	918.4	5.7	13.8	33	2755
sweet-pot	9.6	138	2.7	54	290.7	8.4	5.4	83	1912
peaches	8.5	87	1.7	173	86.8	1.2	4.3	55	57
prunes	12.8	99	2.5	154	85.7	3.9	4.3	65	257
limabeans	17.4	1055	3.7	459	5.1	26.9	38.2	93	
navybeans	26.9	1691	11.4	792		38.4	24.6	217	

Positive Variable x(f) dollars of food f to be purchased daily (dollars)

Free Variable cost total food bill (dollars)

Equations nb(n) nutrient balance (units), cb cost balance (dollars) ;

nb(n).. sum(f, a(f,n)*x(f)) =g= b(n); cb.. cost=e= sum(f, x(f));

Model diet stiglers diet problem / nb,cb /;

Solution of the Diet Problem

Goal: find the cheapest combination of foods that will satisfy the daily requirements of a person

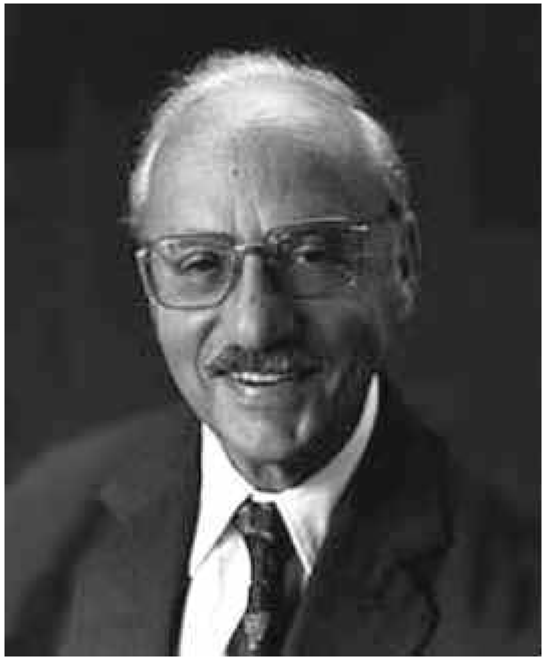
motivated by the army's desire to meet nutritional requirements of the soldiers at minimum cost

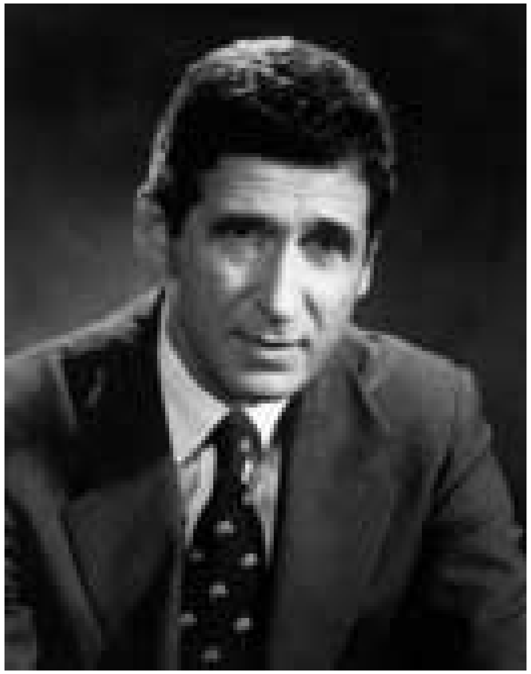
Army's problem had 77 unknowns and 9 constraints.
Stigler solved problem using a heuristic: $39.93/year (1939)
Laderman (1947) used simplex: $39.69/year (1939 prices)

→ first "large-scale computation"
took 120 man days on hand operated
desk calculators (10 human "computers")

http://www.mcs.anl.gov/home/otc/Guide/CaseStudies/diet/index.html

George Dantzig and Ralph Gomory





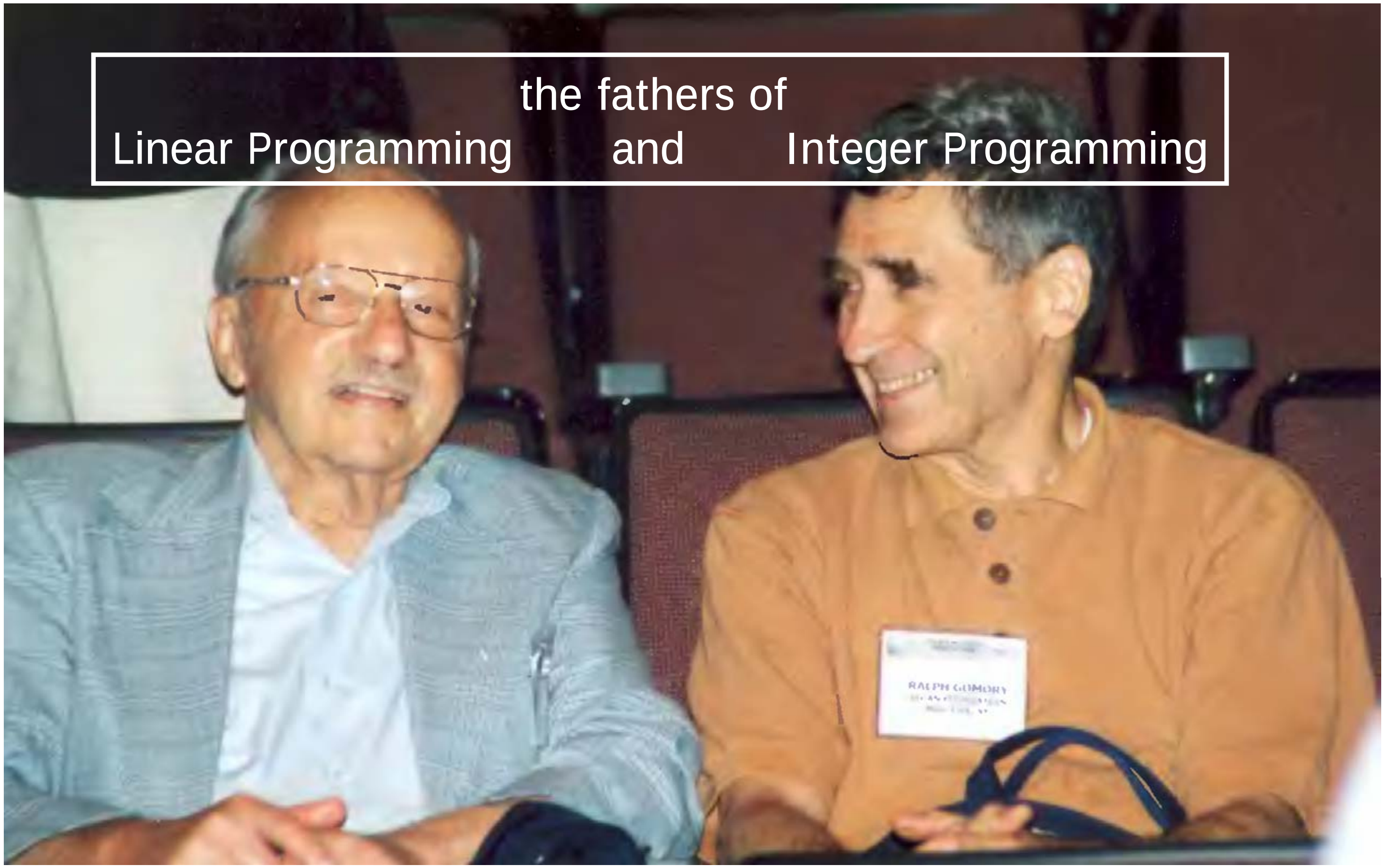
„founding fathers“

~1950
linear programming

~1960
integer programming

George Dantzig and Ralph Gomory

ISMP Atlanta 2000



Dantzig and Bixby

George Dantzig and
Bob Bixby

at the International Symposium on Mathematical Programming,

Atlanta, August 2000

special „simple“ combinatorial optimization problems

Finding a

minimum spanning tree

shortest path

maximum matching

maximal flow through a network

cost-minimal flow

...

solvable in polynomial time by special purpose algorithms

Optimizers' dream: Duality theorems

Max-Flow Min-Cut Theorem

The maximal (s,t)-flow in a capacitated network is equal to the minimal capacity of an (s,t)-cut.

The Duality Theorem of linear programming

$$\begin{array}{c} \max c^T x \\ Ax \le b \\ x \ge 0 \end{array} \quad = \quad \begin{array}{c} \min y^T b \\ y^T A \ge c^T \\ y \ge 0 \end{array}$$

special „hard“ combinatorial optimization problems

travelling salesman problem

location und routing

set-packing, partitioning, -covering

max-cut

linear ordering

scheduling (with a few exceptions)

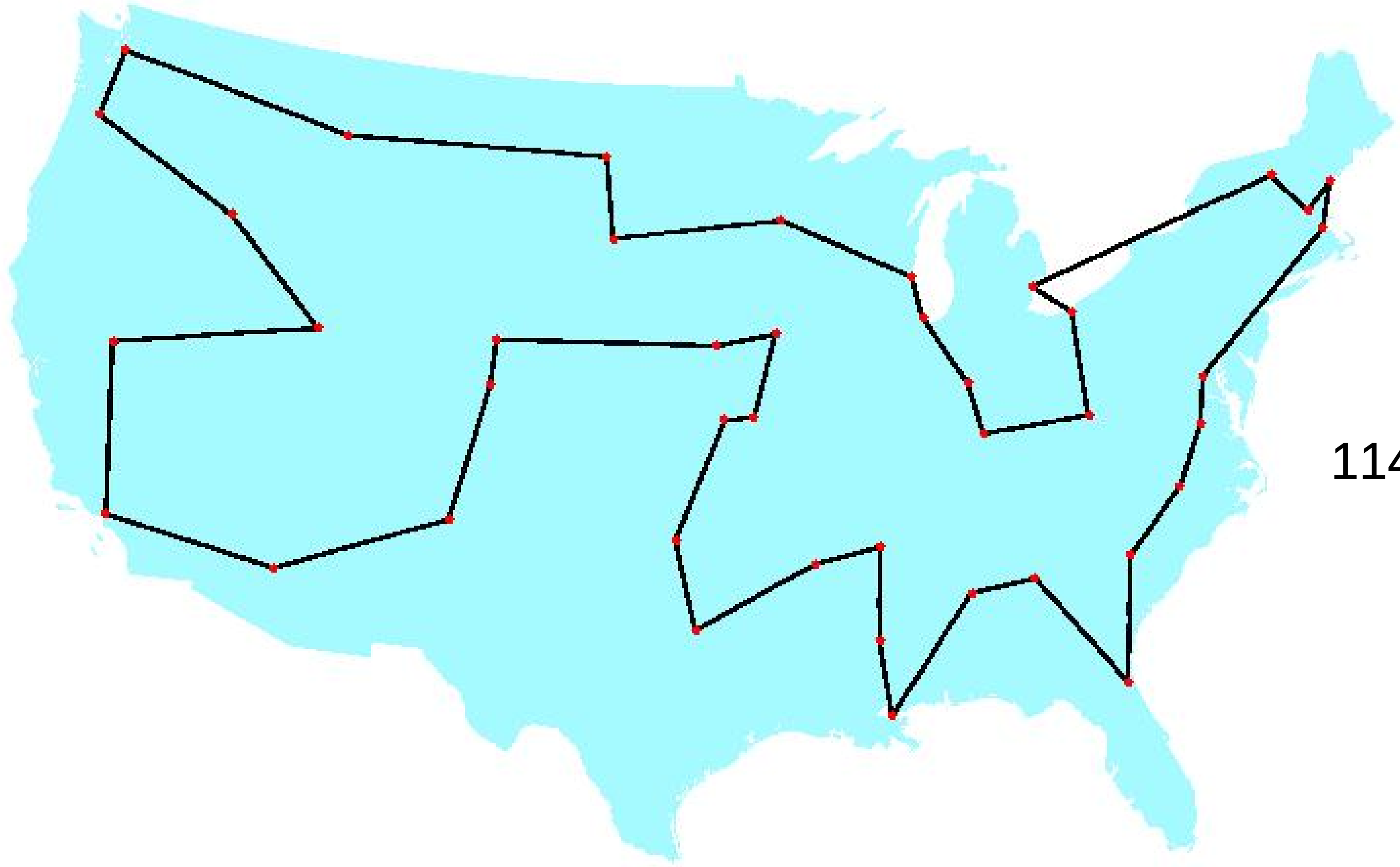
node and edge colouring

...

NP-hard (in the sense of complexity theory)

The most successful solution techniques employ linear programming.

1954, the Beginning of IP
G. Dantzig, D.R. Fulkerson, S. Johnson



USA
49 cities
1146 variables

1954

The Simplex Method

Dantzig, 1947: primal Simplex Method

Lemke, 1954; Beale, 1954: dual Simplex Method

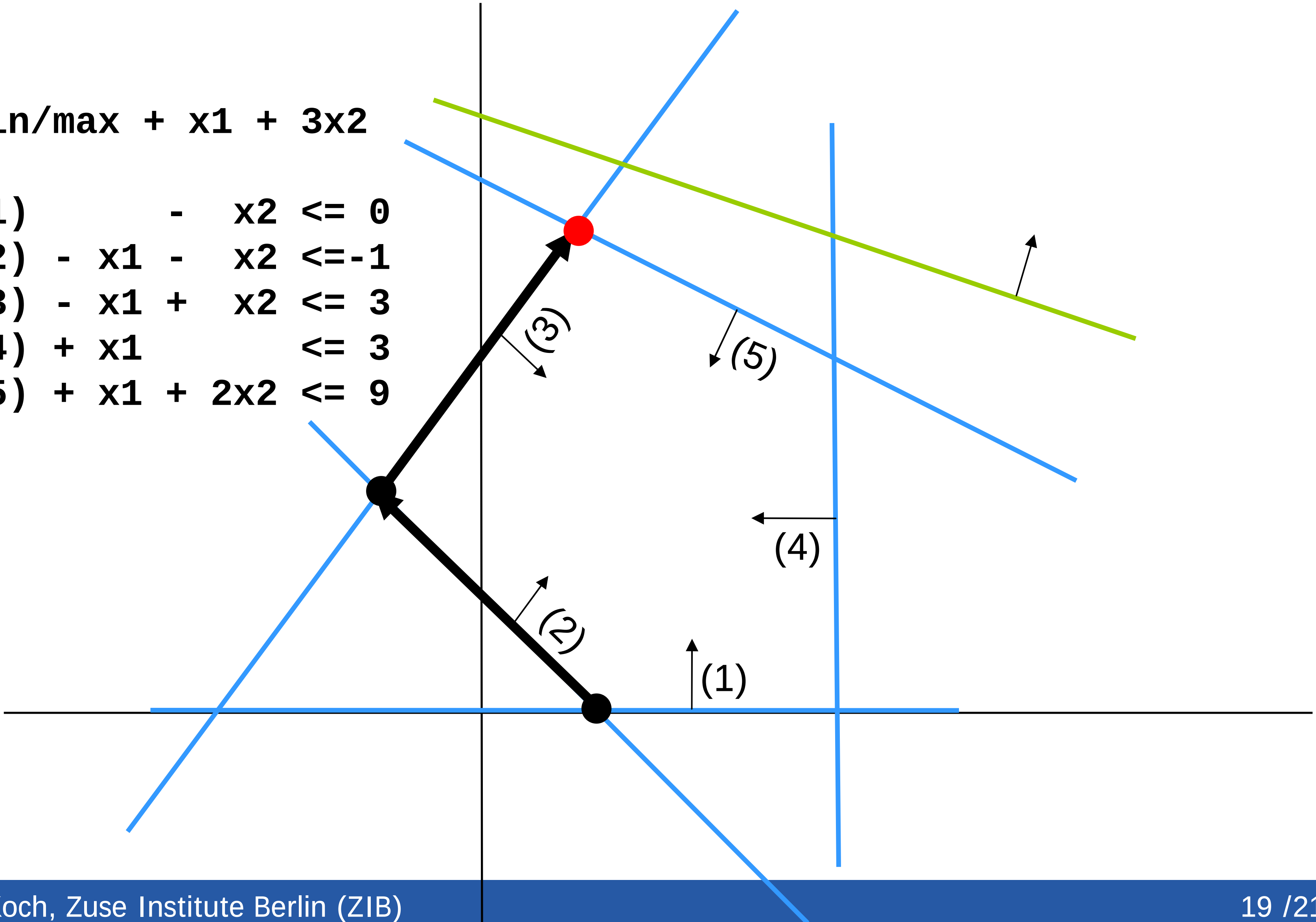
Dantzig, 1953: revised Simplex Method

....

Underlying Idea: Find a vertex of the set of feasible LP solutions (polyhedron) and move to a better neighbouring vertex, if possible.

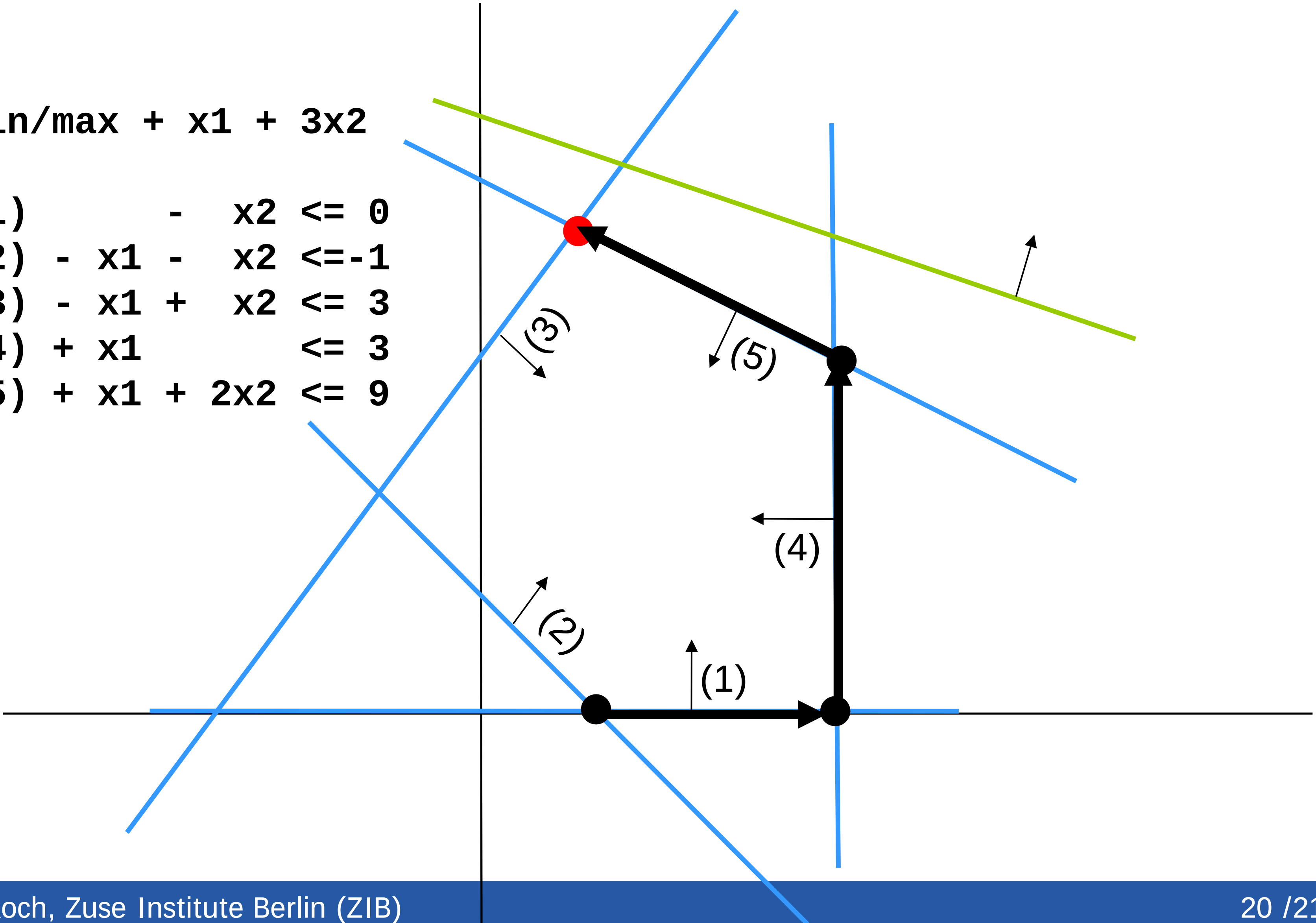
The Simplex Method: an example

```
min/max + x1 + 3x2

(1)        -   x2 <= 0
(2) - x1 -   x2 <=-1
(3) - x1 +   x2 <= 3
(4) + x1         <= 3
(5) + x1 + 2x2 <= 9
```



The Simplex Method: an example

```
min/max + x1 + 3x2

(1)        -   x2 <= 0
(2) - x1 -   x2 <=-1
(3) - x1 +   x2 <= 3
(4) + x1         <= 3
(5) + x1 + 2x2 <= 9
```



computationally important idea of the Simplex Method

Let a (m,n)-Matrix A with full row rank m, an m-vector b and an n-vector c with m<n be given. For every vertex y of the polyhedron of feasible solutions of the LP,

$$\max c^T x$$
$$Ax = b$$
$$x \geq 0$$

$$A = \begin{array}{|c|c|}\hline B & \qquad N \qquad \\ \hline\end{array}$$

there is a non-singular (m,m)-submatrix B (called basis) of A representing the vertex y (basic solution) as follows

$$y_B = B^{-1}b, \quad y_N = 0$$

Many computational consequences:
Update-formulas, reduced cost calculations,
number of non-zeros of a vertex,...

cutting plane technique for integer and mixed-integer programming

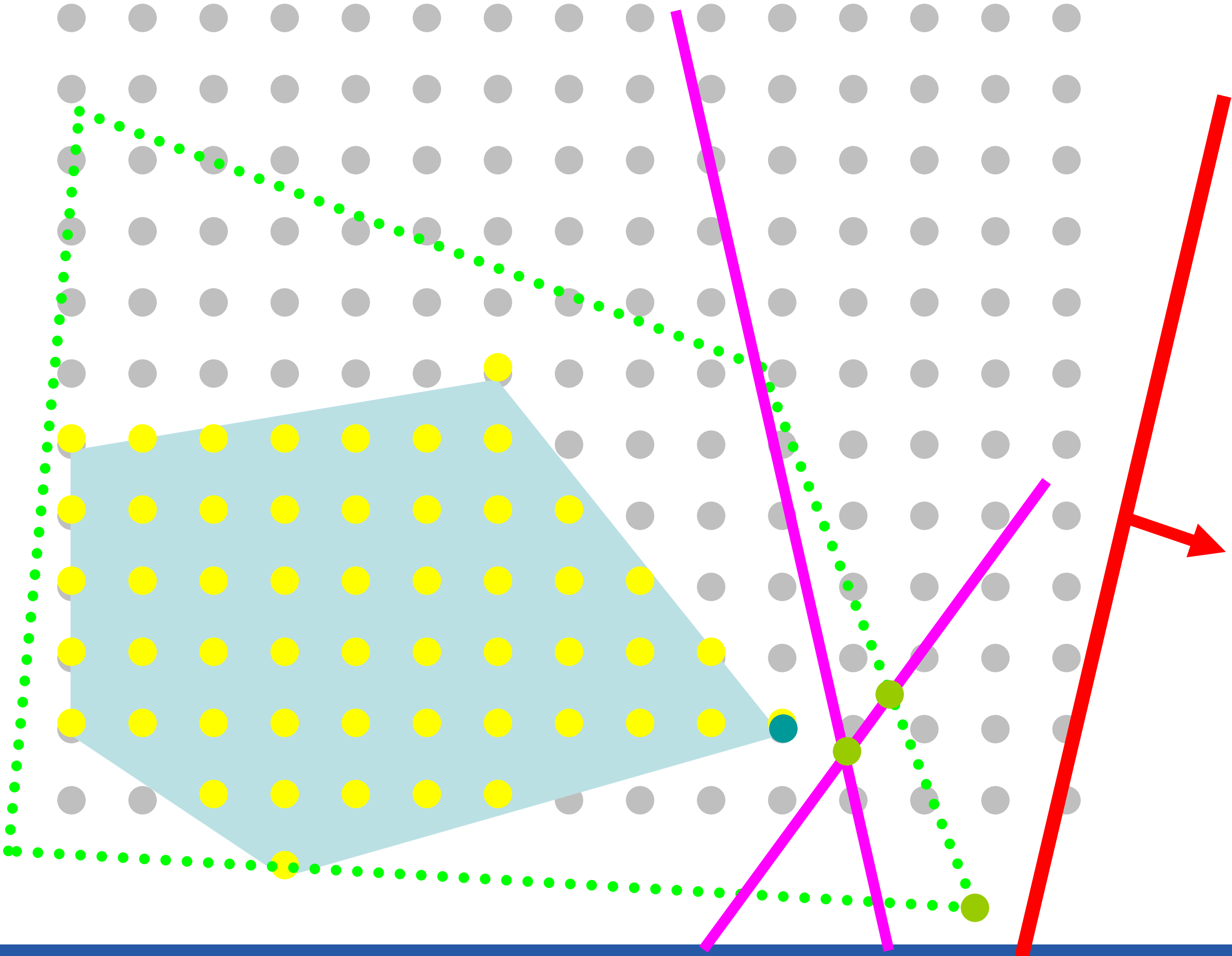
Feasible integer solutions

Objective function

Convex hull

LP-based relaxation

Cutting planes



Which LP solvers are used in practice?

Fourier-Motzkin: hopeless

Ellipsoid Method: total failure

primal Simplex Method: good

dual Simplex Method: better

Barrier Method: for large LPs frequently even better

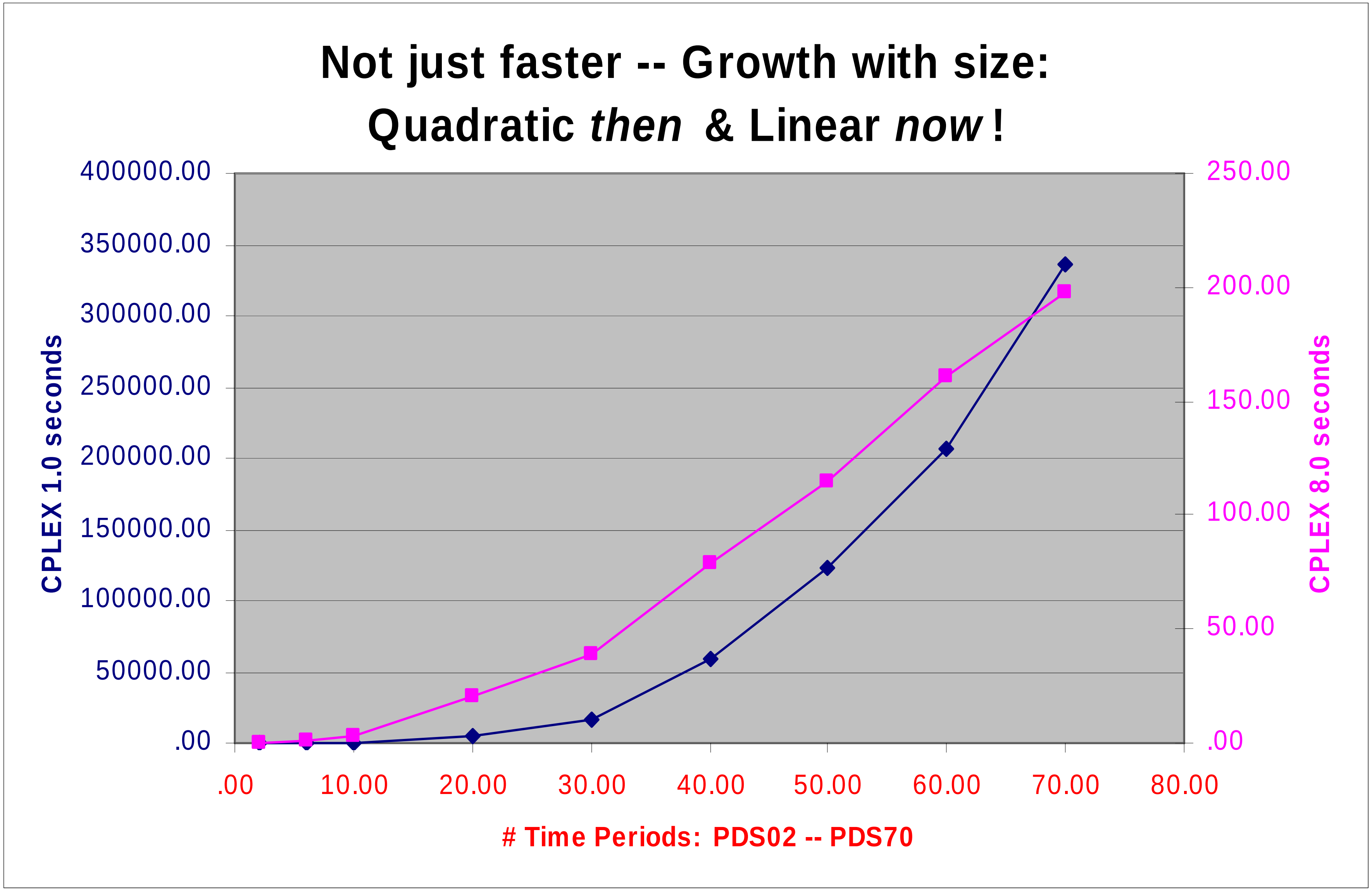
For LP relaxations of MIPs: dual Simplex Method

PDS Models, Courtesy Bob Bixby

“Patient Distribution System”: Carolan, Hill, Kennington, Niemi, Wichmann, *An empirical evaluation of the KORBX algorithms for military airlift applications*, Operations Research **38** (1990), pp. 240-248

MODEL	ROWS	CPLEX1.0 1988	CPLEX5.0 1997	CPLEX8.0 2002	SPEEDUP 1.0→8.0
pds02	2953	**0.4**	**0.1**	**0.1**	**4.0**
pds06	9881	**26.4**	**2.4**	**0.9**	**29.3**
pds10	16558	**208.9**	**13.0**	**2.6**	**80.3**
pds20	33874	**5268.8**	**232.6**	**20.9**	**247.3**
pds30	49944	**15891.9**	**1154.9**	**39.1**	**406.4**
pds40	66844	**58920.3**	**2816.8**	**79.3**	**743.0**
pds50	83060	**122195.9**	**8510.9**	**114.6**	**1066.3**
pds60	99431	**205798.3**	**7442.6**	**160.5**	**1282.2**
pds70	114944	**335292.1**	**21120.4**	**197.8**	**1695.1**
		Primal Simplex	Dual Simplex	Dual Simplex	

Courtesy Bob Bixby



Computational history: 1950 – 1998

1954 Dantzig, Fulkerson, S. Johnson: 42-city TSP

- Solved to optimality using cutting planes and LP

1957 Gomory

- Cutting plane algorithm: A complete solution

1960 Land, Doig; 1965 Dakin

- Branch-and-bound (B&B)

1971 MPSX/370, Benichou et al.

1972 UMPIRE, Forrest, Hirst, Tomlin (**and** Beale)

1972 – 1998 Good B&B remained the state-of-the-art in commercial codes, in spite of

- 1973 Padberg
- 1974 Balas (disjunctive programming)
- 1983 Crowder, Johnson, Padberg: PIPX, pure 0/1 MIP
- 1987 Van Roy and Wolsey: MPSARX, mixed 0/1 MIP
- Grötschel, Padberg, Rinaldi ...TSP (120, 666, 2392 city models solved)

1998 … A New Generation of MIP Codes

Linear programming

- Stable, robust dual simplex

Variable / node selection

- Influenced by traveling salesman problem

Heuristics

- 11 different tried at root
- Retried based upon success

Presolve

- Numerous small ideas

Cutting planes

- Gomory, knapsack covers, flow covers, mix-integer rounding, cliques, GUB covers, implied bounds, path cuts

Solving MIPs

Branching

- Decompose problem in smaller subproblems
- Solve subproblems recursively $\rightarrow$ branching tree

Bounding

- LP relaxation $\rightarrow$ lower bound on objective function

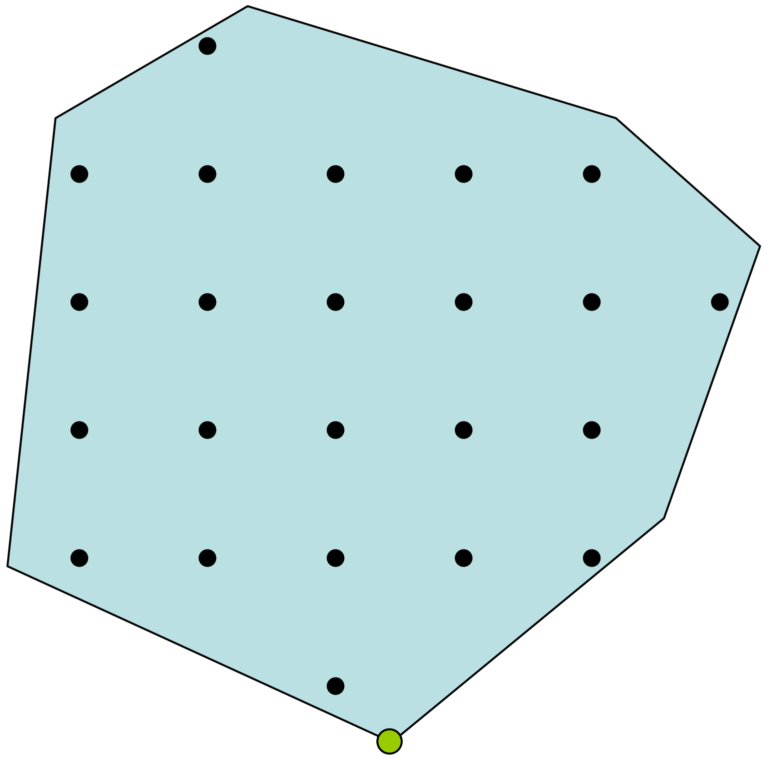
Cutting planes

- Tighten LP relaxation $\rightarrow$ improved lower bounds

Column generation

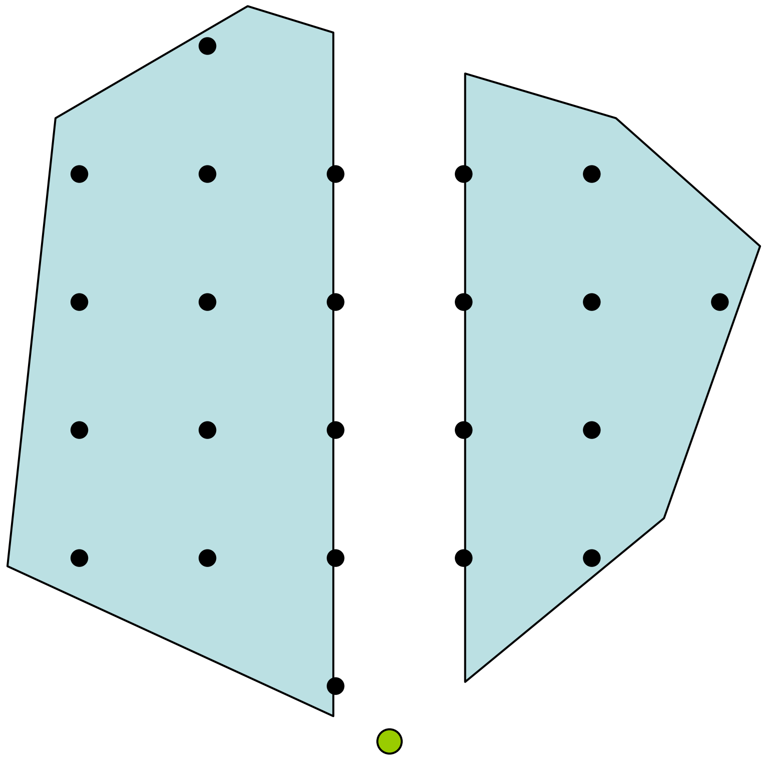
- Dynamic generation of problem variables

Current solution is infeasible

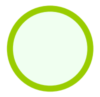


Branching (MIP + CP + SAT)

Decomposition into subproblems removes infeasible solution

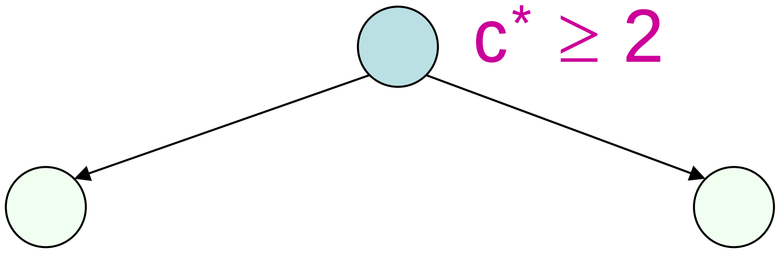


Bounding (MIP)



Root node defines global problem

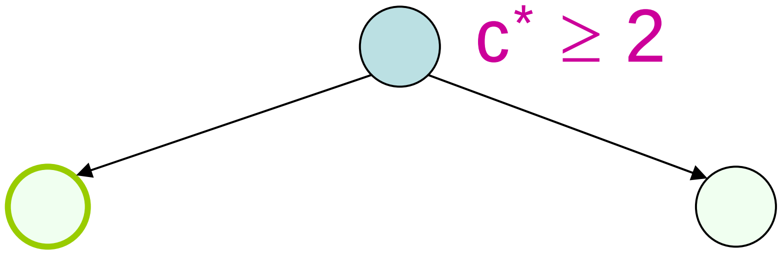
Bounding (MIP)



LP relaxation yields lower bound

Branching decomposes problem into subproblems

Bounding (MIP)

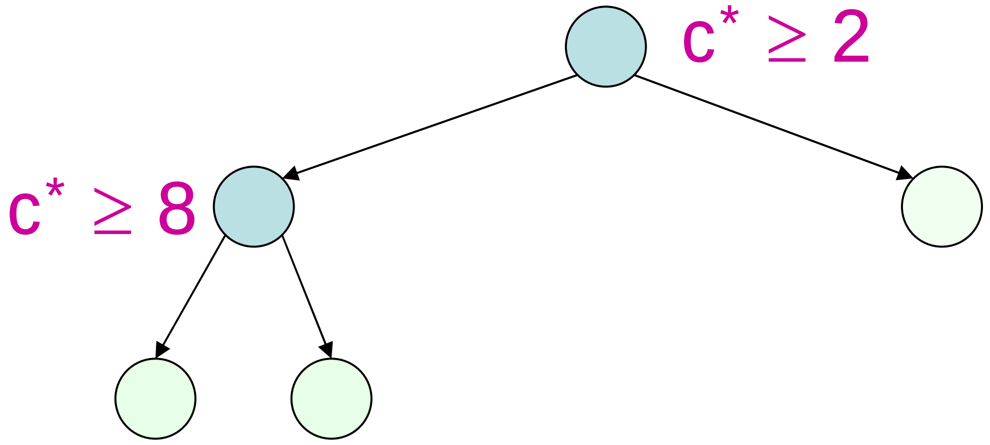


LP relaxation yields lower bound

Branching decomposes problem into subproblems

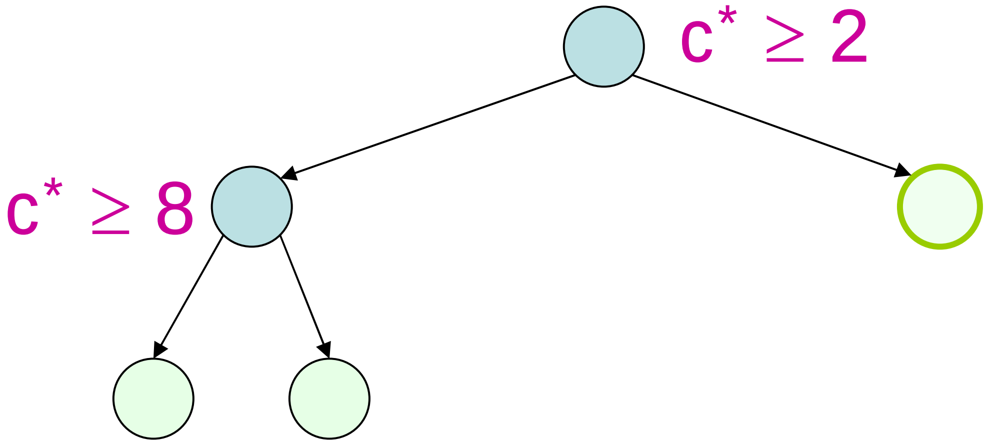
LP relaxation is solved for subproblems

Bounding (MIP)



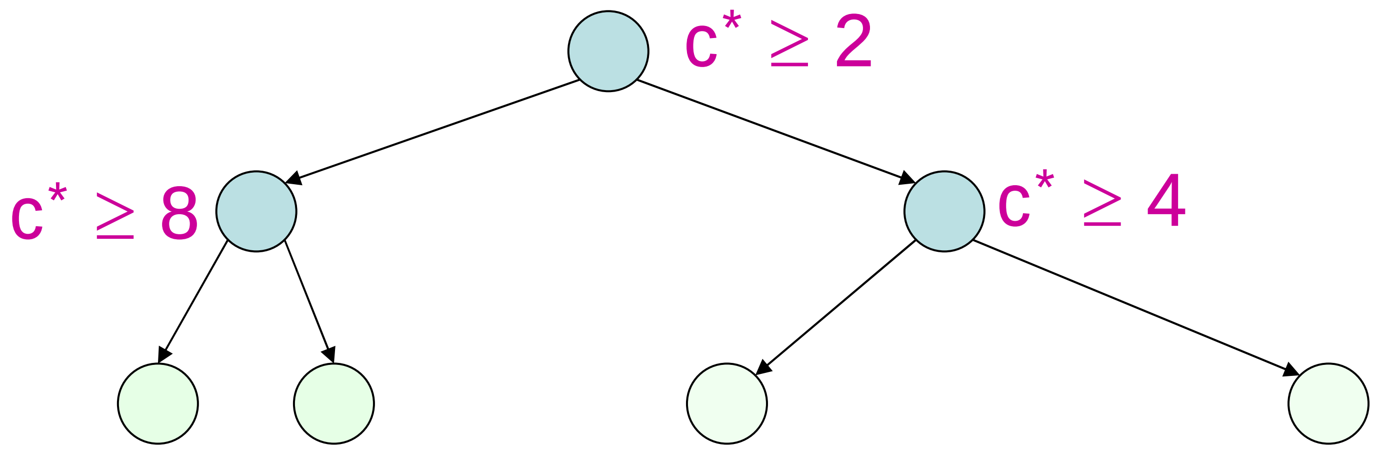
LP relaxation yields lower bounds

Bounding (MIP)



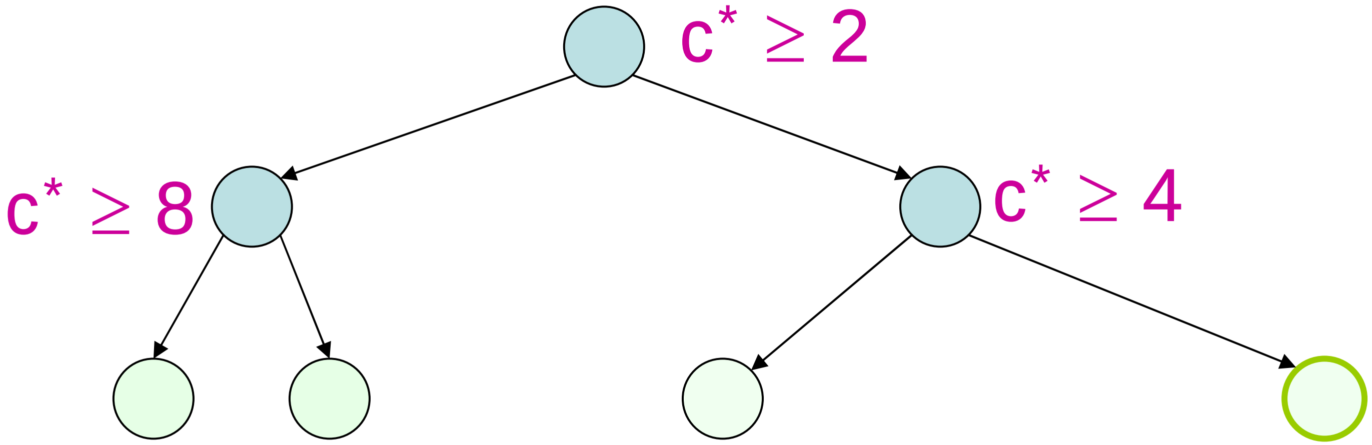
LP relaxation yields lower bounds

Bounding (MIP)



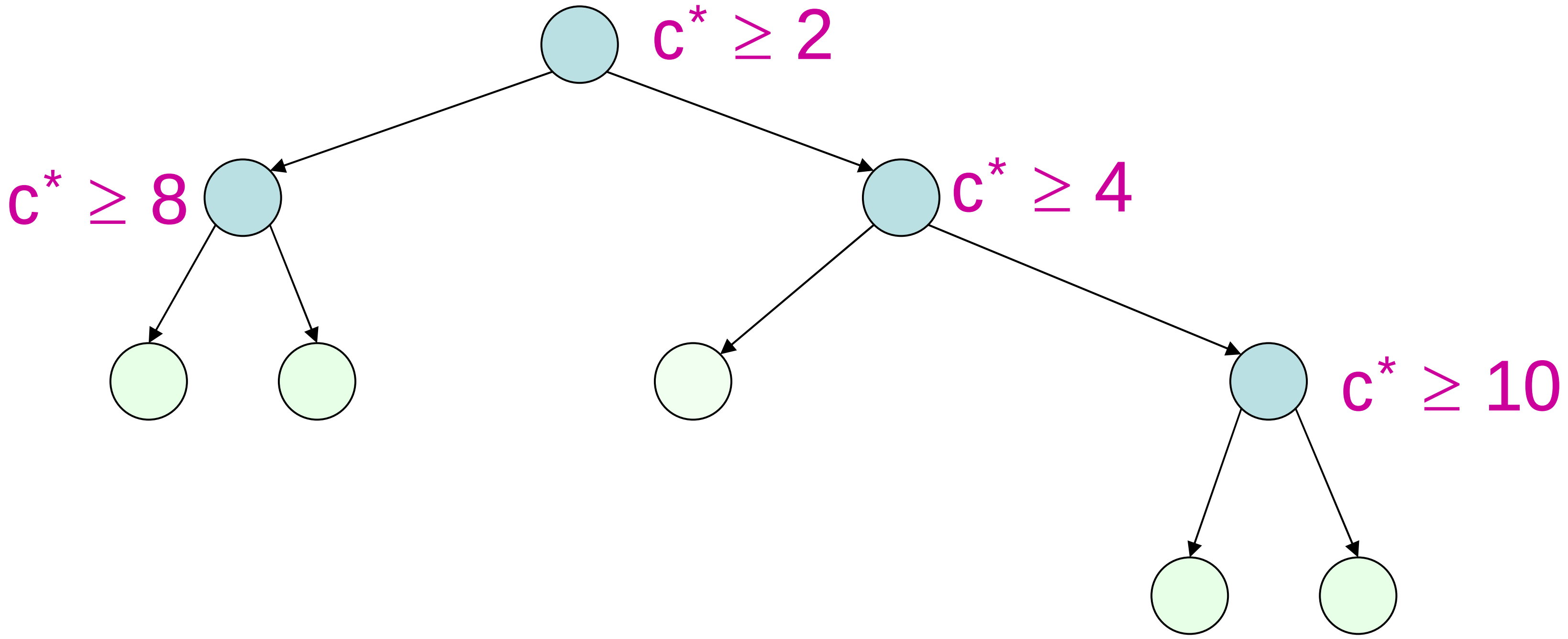
LP relaxation yields lower bounds

Bounding (MIP)



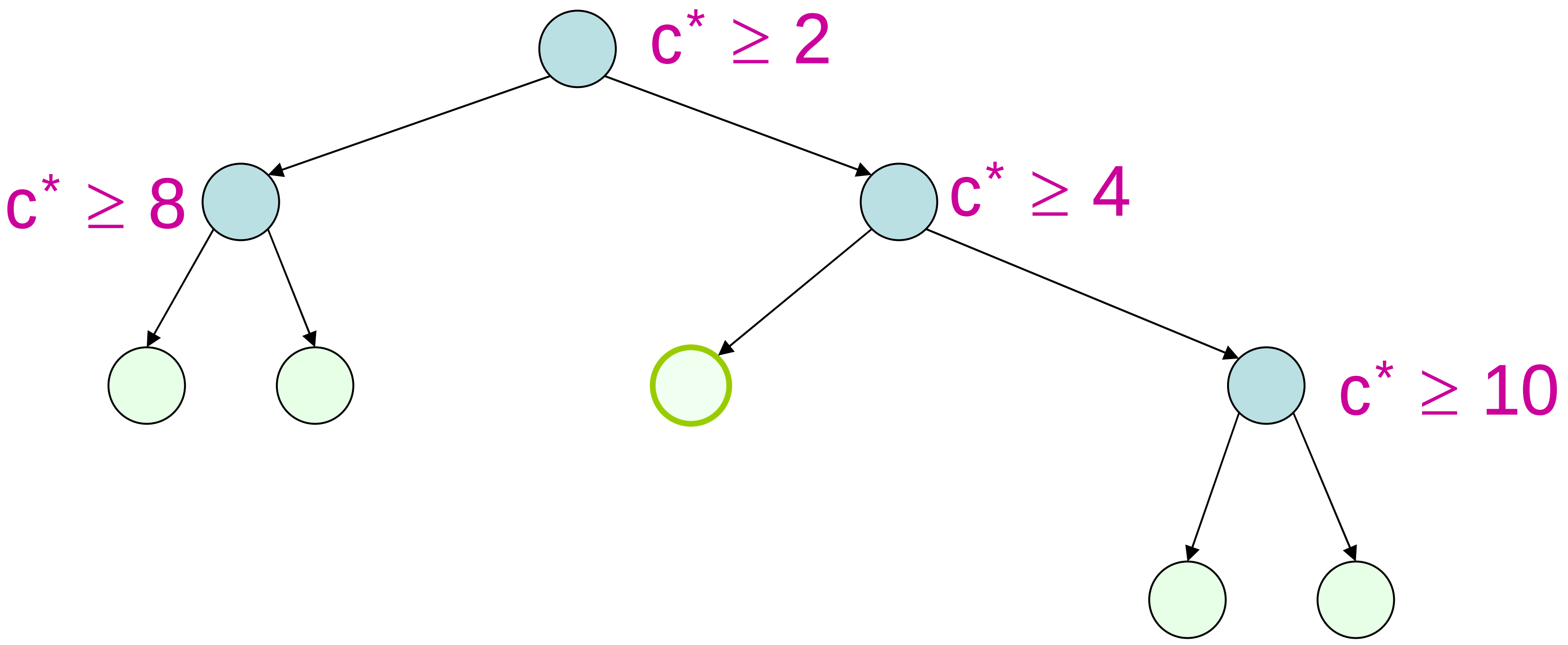
LP relaxation yields lower bounds

Bounding (MIP)



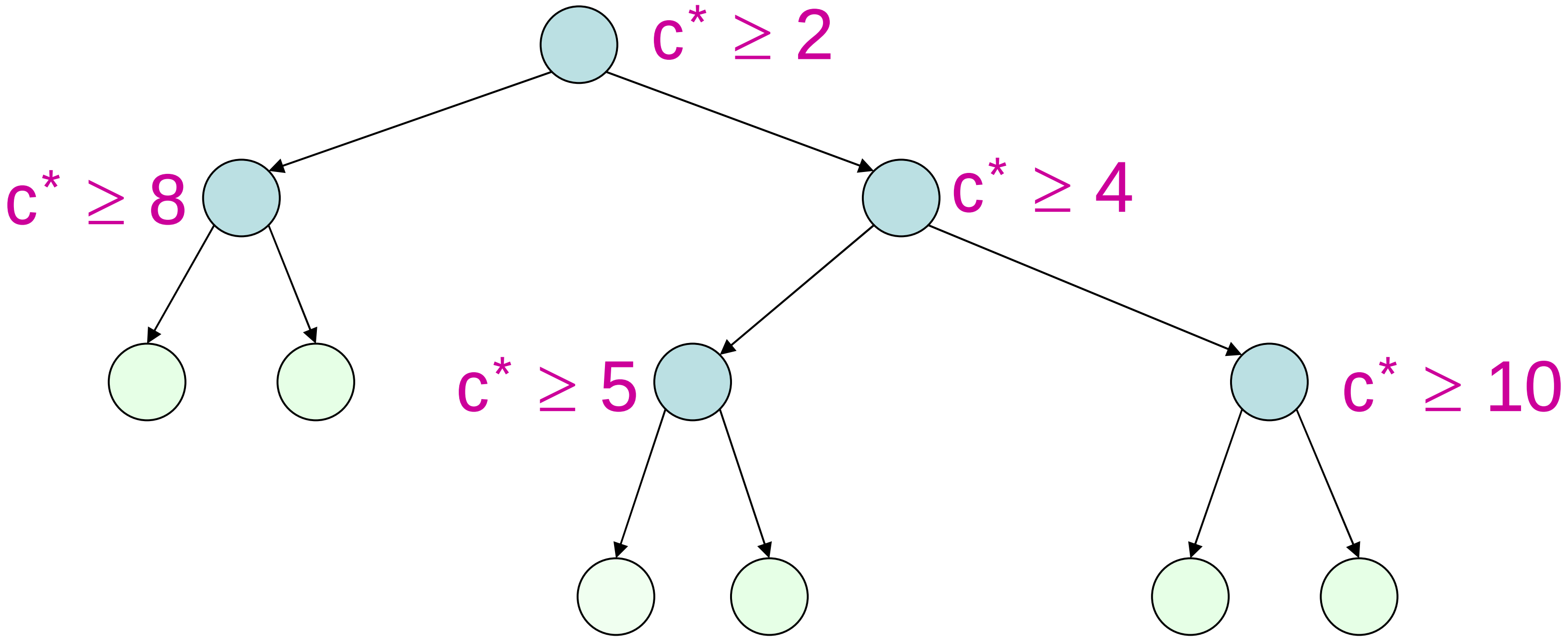
LP relaxation yields lower bounds

Bounding (MIP)

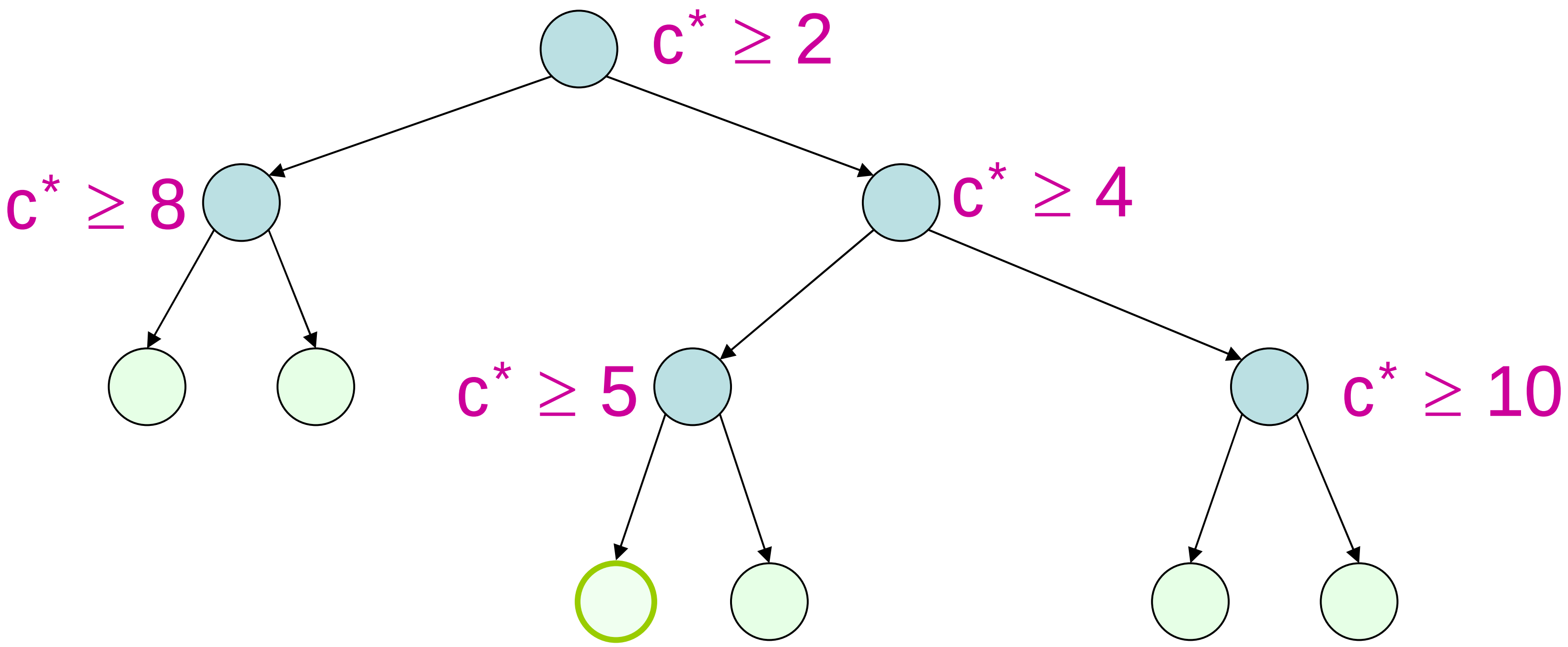


LP relaxation yields lower bounds

Bounding (MIP)

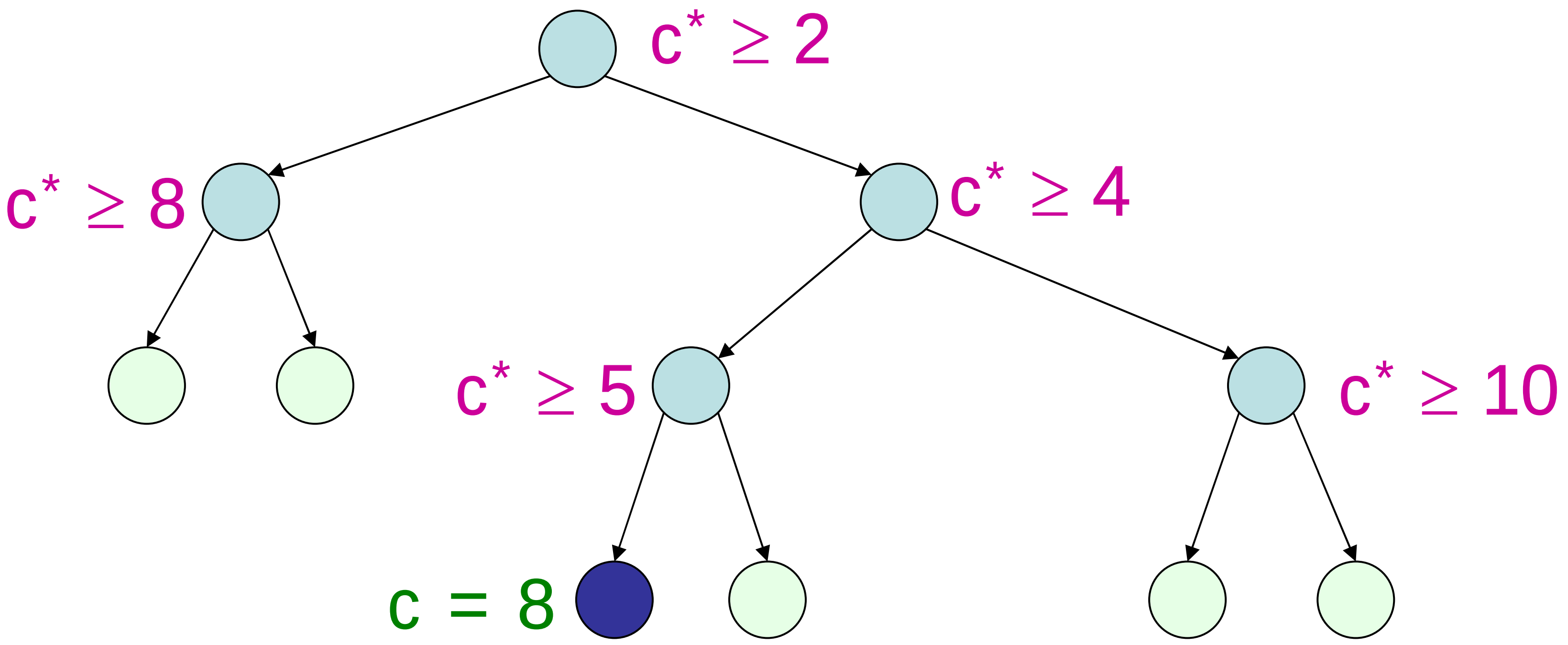


LP relaxation yields lower bounds



LP relaxation yields lower bounds

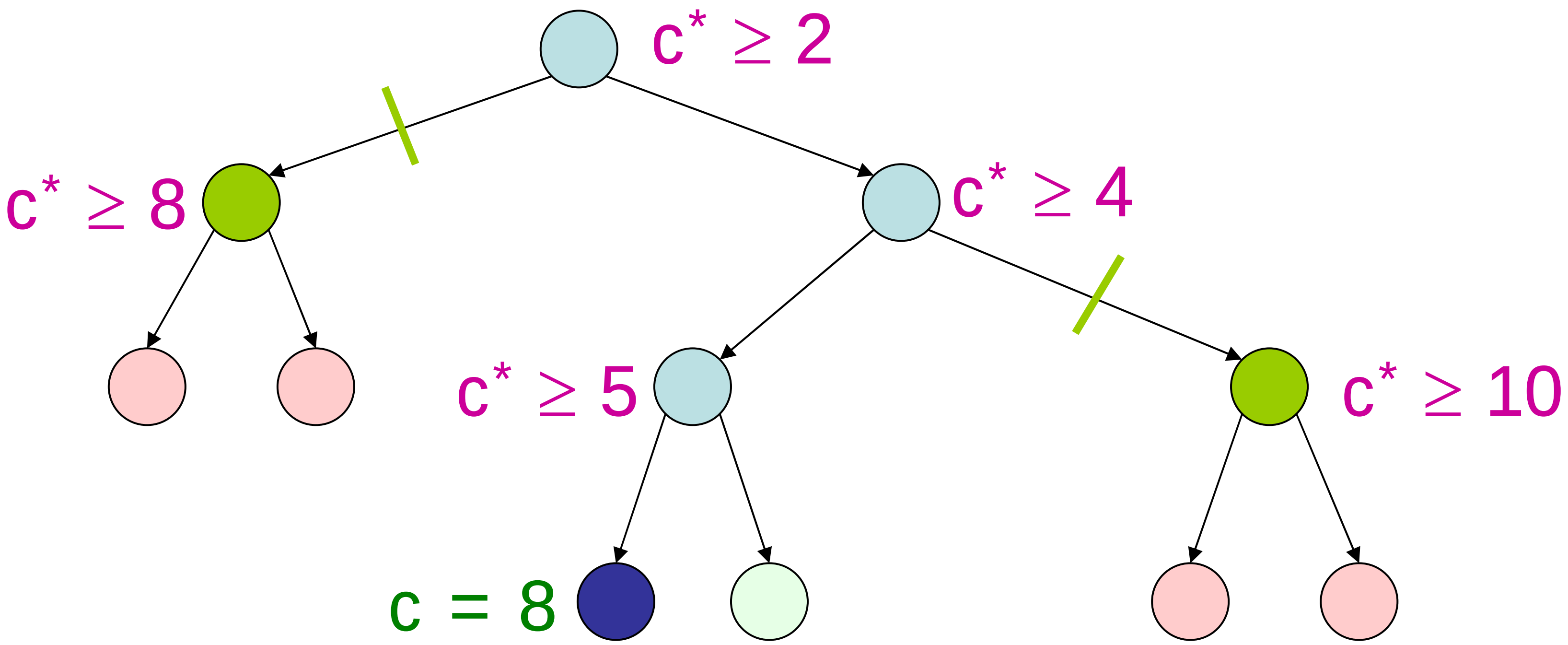
Bounding (MIP)



LP relaxation yields lower bounds

Primal solution yields upper bound

Bounding (MIP)



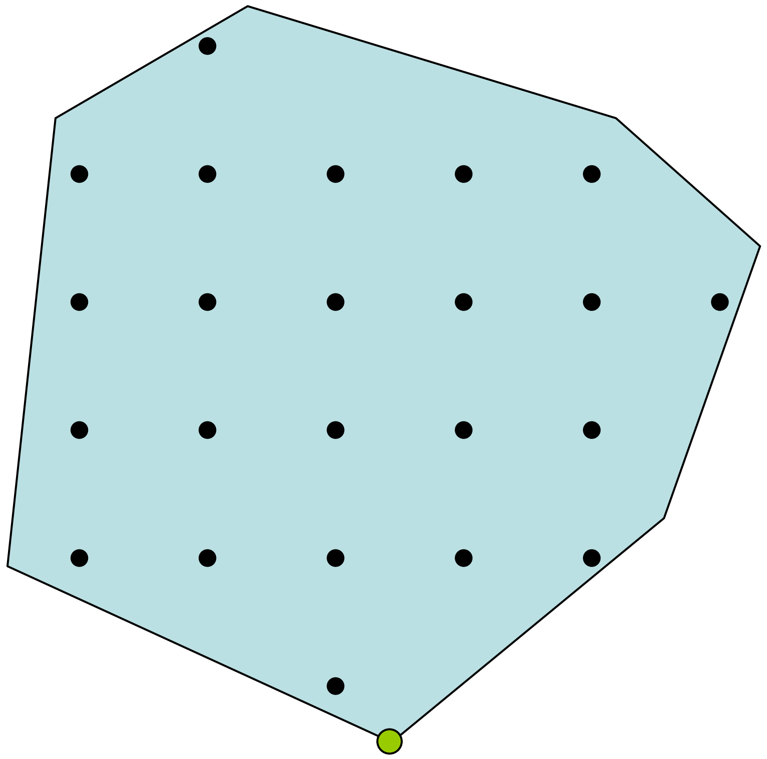
LP relaxation yields lower bounds

Primal solution yields upper bound

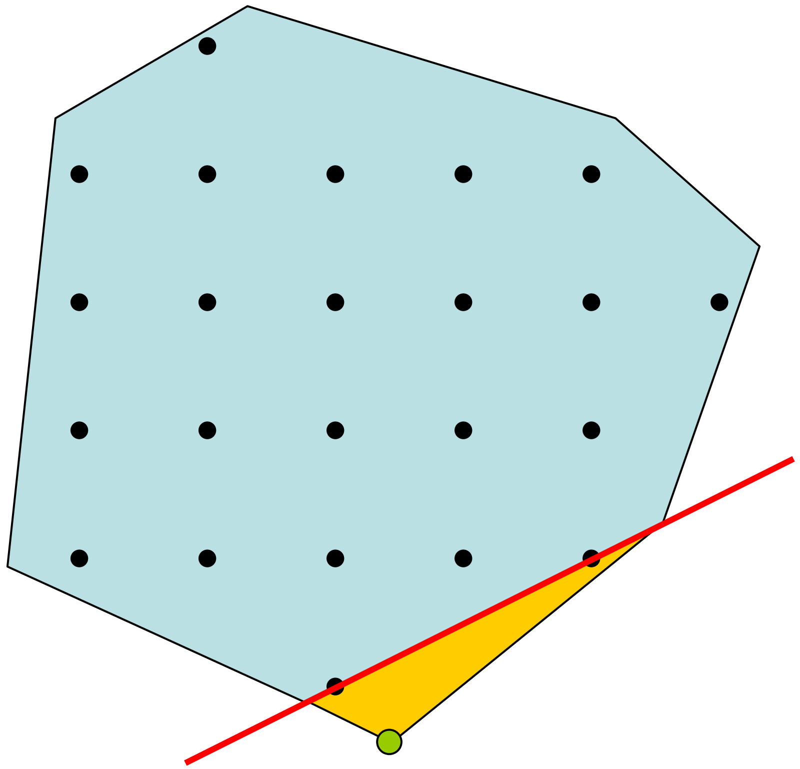
Subproblems cannot contain better solution

Cutting Planes (MIP)

Current solution is infeasible



Infeasible solution is separated by cutting plane



Cutting Planes – Example (MIP)

Binary knapsack problem:

$$\begin{array}{ll} \max & 3x_1 + 5x_2 + 5x_3 + 4x_4 + x_5 \\ \text{s.t.} & 3x_1 + 6x_2 + 7x_3 + 6x_4 + 2x_5 \leq 18 \\ & x_1, x_2, x_3, x_4, x_5 \in \{0,1\} \end{array}$$

LP solution:

$$x_1 = x_2 = x_3 = 1,\ x_4 = 1/3,\ x_5 = 0$$

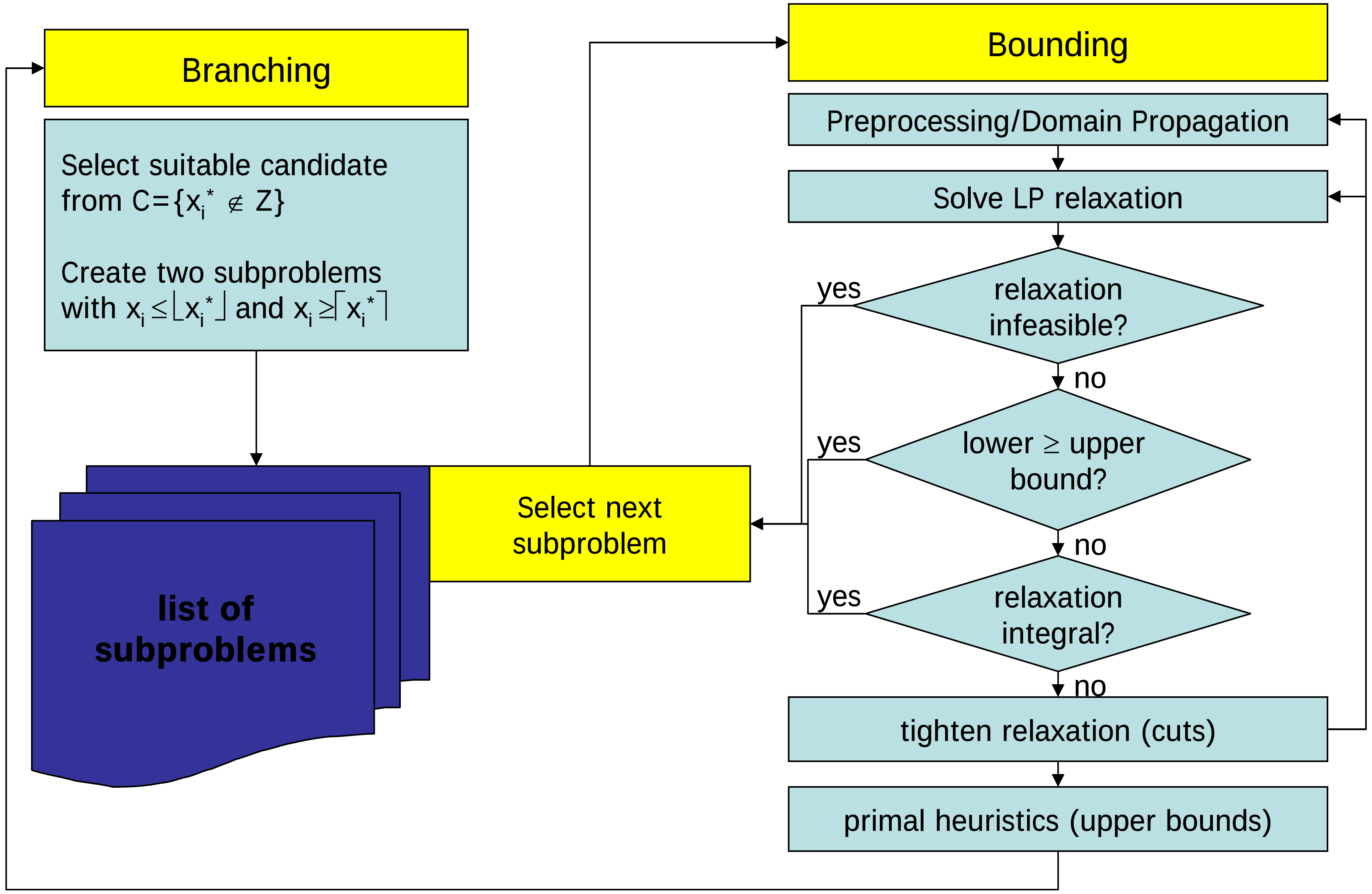
Knapsack cover cut:

$$x_2 + x_3 + x_4 \leq 2$$

New LP solution:

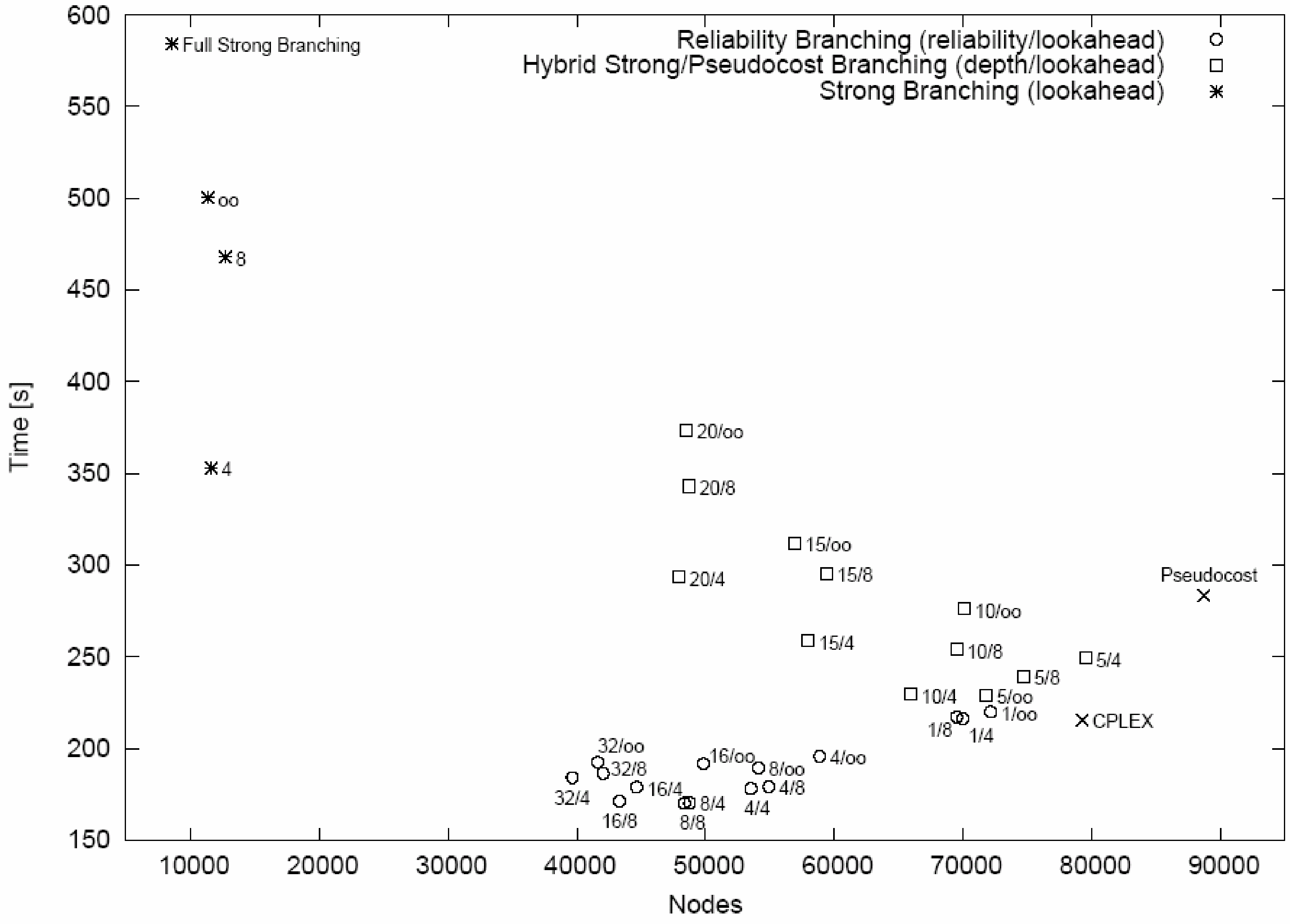
$$x_1 = x_2 = x_3 = 1,\ x_4 = 0,\ x_5 = 1$$

Branch-and-Bound

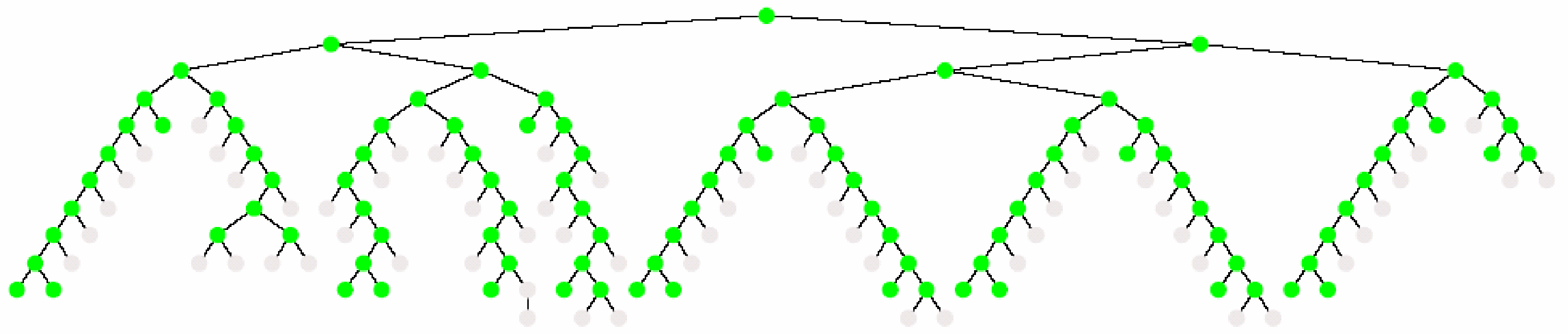


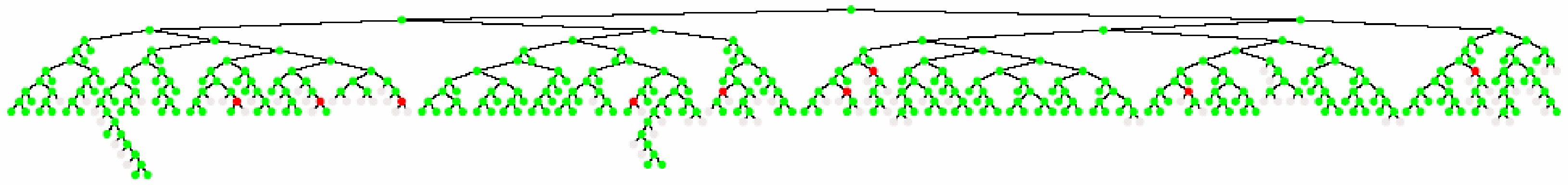
Branching
Select suitable candidate from $C=\{x_i^* \notin Z\}$
Create two subproblems with $x_i \le \lfloor x_i^* \rfloor$ and $x_i \ge \lceil x_i^* \rceil$
list of subproblems
Select next subproblem
Bounding
Preprocessing/Domain Propagation
Solve LP relaxation
relaxation infeasible?
yes
no
lower $\ge$ upper bound?
yes
no
relaxation integral?
yes
no
tighten relaxation (cuts)
primal heuristics (upper bounds)

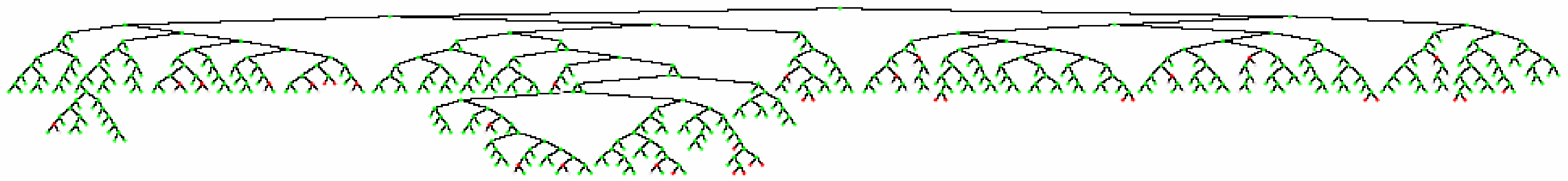
Branching Algorithms



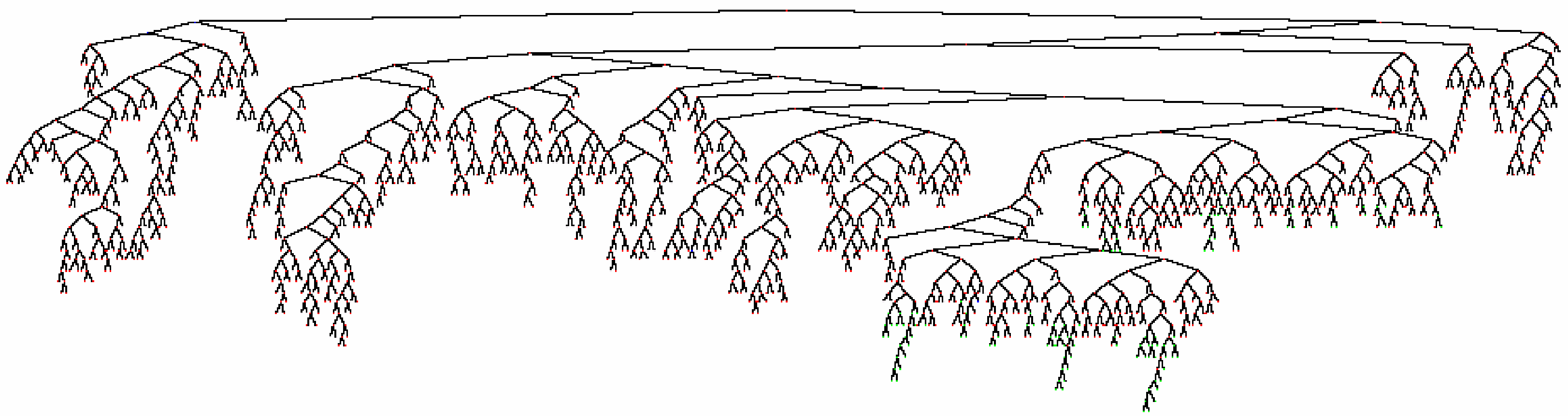
Branching Trees

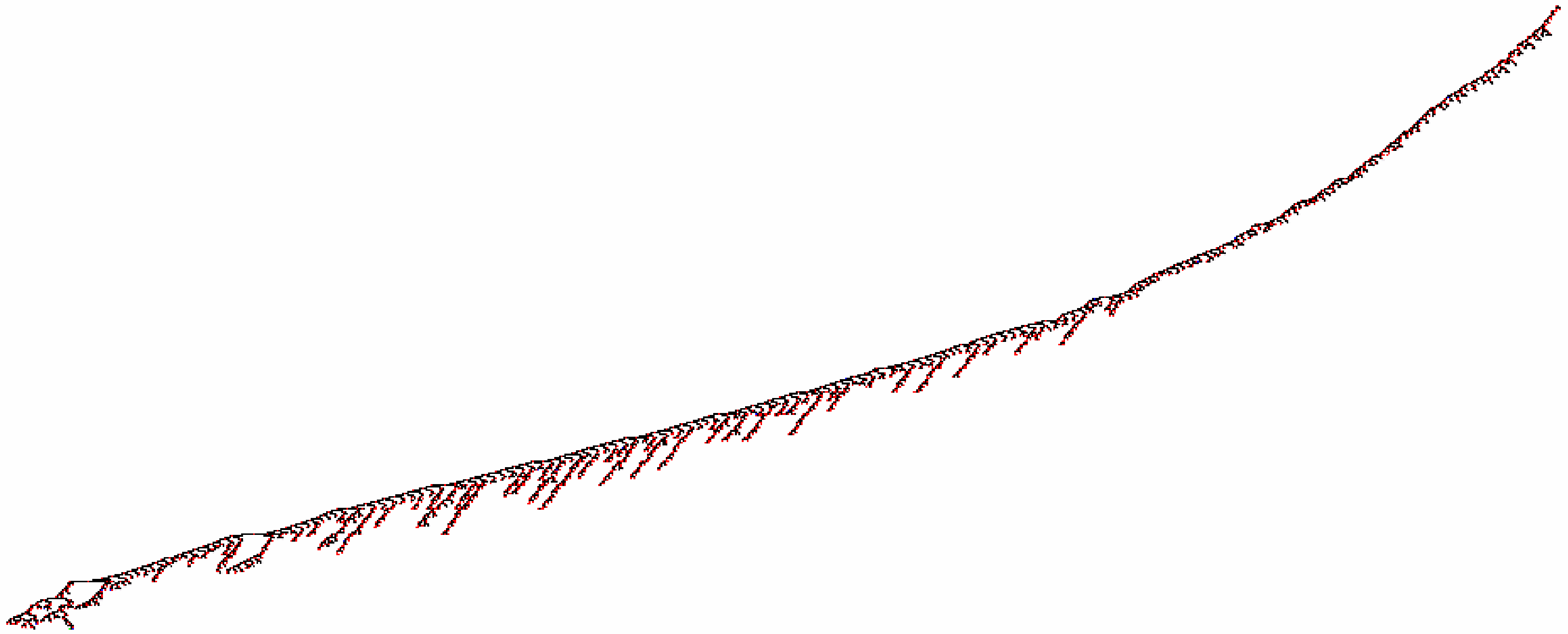






Branching Trees





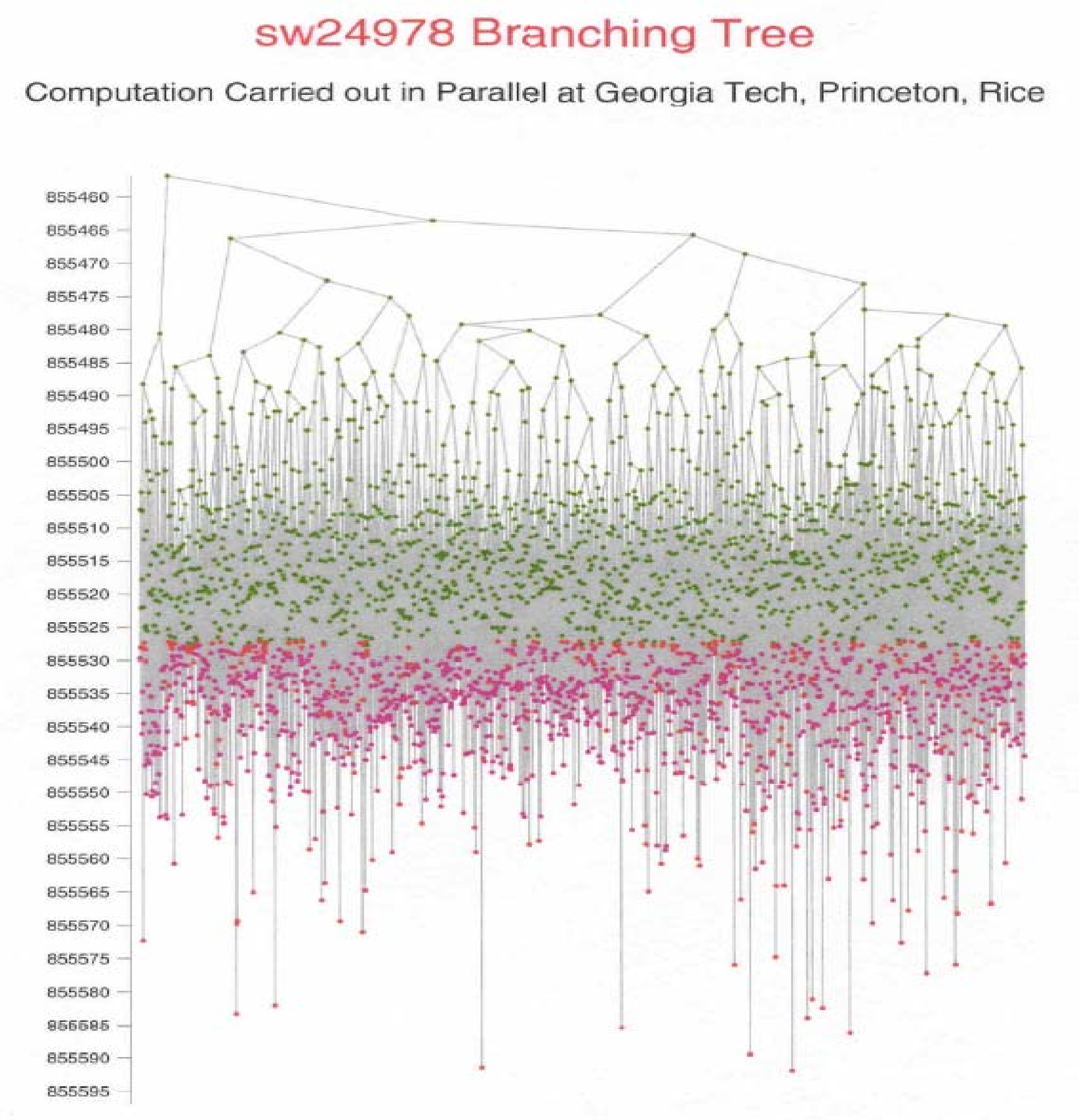
A Branching Tree

Applegate

Bixby

Chvátal

Cook



LP as subroutine of MIP

The Simplex algorithm is the core engine of every Branch-and-Bound based MIP solver

At least one LP is solved at each node in the branching tree, sometimes even more than 10.

SoPlex is a composite Simplex.

It features primal and dual algorithms for column and row basis.

Why some MIPs are difficult

- Genuine bad formulation, e. g. Sudoku with integer variables.
- LP is difficult (e. g. stp3d)
- Bad numerical properties (e. g. momentum3)
- Difficult to find primal solution (e. g. stp3d)
- Bad dual bound (e. g. $x + y \leqslant 1 + z$)
- Just big
- Nobody knows

Experienced model builders do not use models/constructs that are likely not to work well. Software is tailored towards models that are used by experienced model builders.

TSP model

$$G = (V, A, c), \quad A = V \times V, \quad c_a > 0, a \in A$$

$$\min \sum_{(i,j)\in A} c_{ij} y_{ij}$$

subject to

$$u_i - u_j + (n-1) y_{ij} \leqslant n - 2 \quad \text{for all } (i,j) \in A, j \neq 1 \tag{1}$$

$$\sum_{(j,i)\in\delta^-(i)} y_{ji} = 1 \quad \text{for all } i \in V \tag{2}$$

$$\sum_{(i,j)\in\delta^+(i)} y_{ij} = 1 \quad \text{for all } i \in V \tag{3}$$

$$u_1 = 0 \tag{4}$$

$$1 \leqslant u_i \leqslant n - 1 \quad \text{for all } i \in \{2, \ldots, n\} \tag{5}$$

$$y_{ij} \in \{0, 1\} \quad \text{for all } (i,j) \in A \tag{6}$$

Extension

M. van Vyve and L. A. Wolsey describe in *Approximate Extended Formulations* an extension of the model. $V_l \subset V$, for all $l \in V$ are subsets containing the n nearest vertices to l. The following constraints are SEC for all subtours up to size $n + 1$:

$$\sum_{(i,j)\in\delta^-(j)} w^l_{ij} - \sum_{(j,i)\in\delta^+(j)} w^l_{ji} \geqslant 0 \quad \text{for all } l \in V, j \in V_l, j \neq l \tag{7}$$

$$\sum_{(i,l)\in\delta^-(l)} w^l_{il} - \sum_{(l,i)\in\delta^+(l)} w^l_{li} \geqslant 1 \quad \text{for all } l \in V \tag{8}$$

$$0 \leqslant w^l_{ij} \leqslant y_{ij} \quad \text{for all } l \in V, \quad (i,j) \in A(V_l) \cup \delta^-(V_l) \cup \delta^+(V_l) \tag{9}$$

Strengthened Extension

$$\sum_{(i,j)\in\delta^-(j)} w_{ij}^l - \sum_{\substack{(j,i)\in\delta^+(j)\\ i\in V_l}} w_{ji}^l = 0 \quad \text{for all } l \in V, j \in V_l, j \neq l \tag{10}$$

$$\sum_{(i,l)\in\delta^-(l)} w_{il}^l - \sum_{\substack{(l,i)\in\delta^+(l)\\ i\in V_l}} w_{li}^l = 1 \quad \text{for all } l \in V \tag{11}$$

$$w_{ij}^l \leqslant y_{ij} \quad \text{for all } l \in V, \; (i,j) \in A(V_l) \cup \delta^-(V_l) \tag{12}$$

$$w_{ij}^l \in \{0,1\} \quad \text{for all } l \in V, \; (i,j) \in A(V_l) \cup \delta^-(V_l) \tag{13}$$

$$u_i \in \{1,\ldots,n-1\} \quad \text{for all } i \in \{2,\ldots,n\} \tag{14}$$

Results CPLEX 10 $|V_l|$=13

$\lvert V_l \rvert = 13$	$w \geqslant 0$ $(w \leqslant 1)$			$w \in \{0,1\}$			$u \in \mathbb{N}$ $w \in \{0,1\}$		
	P	B&B	[s]	P	B&B	[s]	P	B&B	[s]
$\geqslant, \delta^+$	A	962	68	A	1,090	141	A	1,667	240
$\geqslant$	A	1,852	90	A	130	40	A	1,239	175
$=, \delta^+$	C	96	90	-	42	237	C	1,476	2,257
$=$	B	56	60	B	1,908	2,394	B	2,385	3,400

ap at optimal solution 0.04-0.10%

implex iterations ≈41,000 for
$v \geqslant 0, \geqslant, \delta^+$ and $w \geqslant 0, =$

bjective
$\sqrt{0.1((x_i - x_j)^2) + (y_i - y_j)^2))}\rceil$

	Rows	Cols	Non-0s
A	10,370	9,791	62,790
B	32,210	31,631	106,470
C	54,050	53,471	171,990

Times for permuted instances

	CPLEX (D)	CPLEX (S)	SCIP (D)
1	21,82	9,79	137,50
2	24,07	15,89	188,43
3	45,01	16,01	185,22
4	50,41	17,15	200,69
5	46,26	17,31	156,29
6	172,74	20,38	149,63
7	216,57	21,17	167,70
8	38,28	23,10	124,06
9	174,49	23,75	177,86
12	159,52	25,26	177,92
13	44,61	25,95	123,89
...	...	...	...
57	351,84	153,00	178,00
58	415,42	153,51	146,45
59	111,33	189,87	119,82
60	181,70	216,81	148,46
Average	161,53	62,54	159,28
Factor	19,04	22,15	2,66

SCIP

constraint integer programming integrates

- constraint programming (CP)
- mixed integer programming (MIP)
- SAT solving techniques

SCIP can be used as

- black box solver for MIP and SAT
- framework for constraint integer programming, including branch-cut-and-price

available in source code

- 215 901 lines of C source code

free to use for non-commercial purposes

SCIP as Standalone MIP Solver

reads MPS or LP file format

directly reads ZIMPL models

built-in MIP specific components:

- branching rules (reliability, strong, most infeasible, ...)
- primal heuristics (rounding, diving, feas. pump, ...)
- node selectors (depth-first, best-first with plunging)
- presolving (dual fixing, probing, ...)
- cut separators (clique, impl. bounds, c-MIR, Gomory, strong CG, lifted knapsack cover)

SCIP as CIP Framework

C interface with C++ wrapper classes

infrastructure to support user plugins

- subproblem and branching tree management
- global cut pool and cut selection
- pricing column management
- event mechanism (bound changes, new solutions, ...)
- lots of statistical data about the solving process
- efficient memory management

all existing MIP components are implemented as user plugins
$\Rightarrow$ interface is powerful enough for most applications

Benchmarks (commercial)

MIPLIB2003 / Mittelmann / Lodi test set

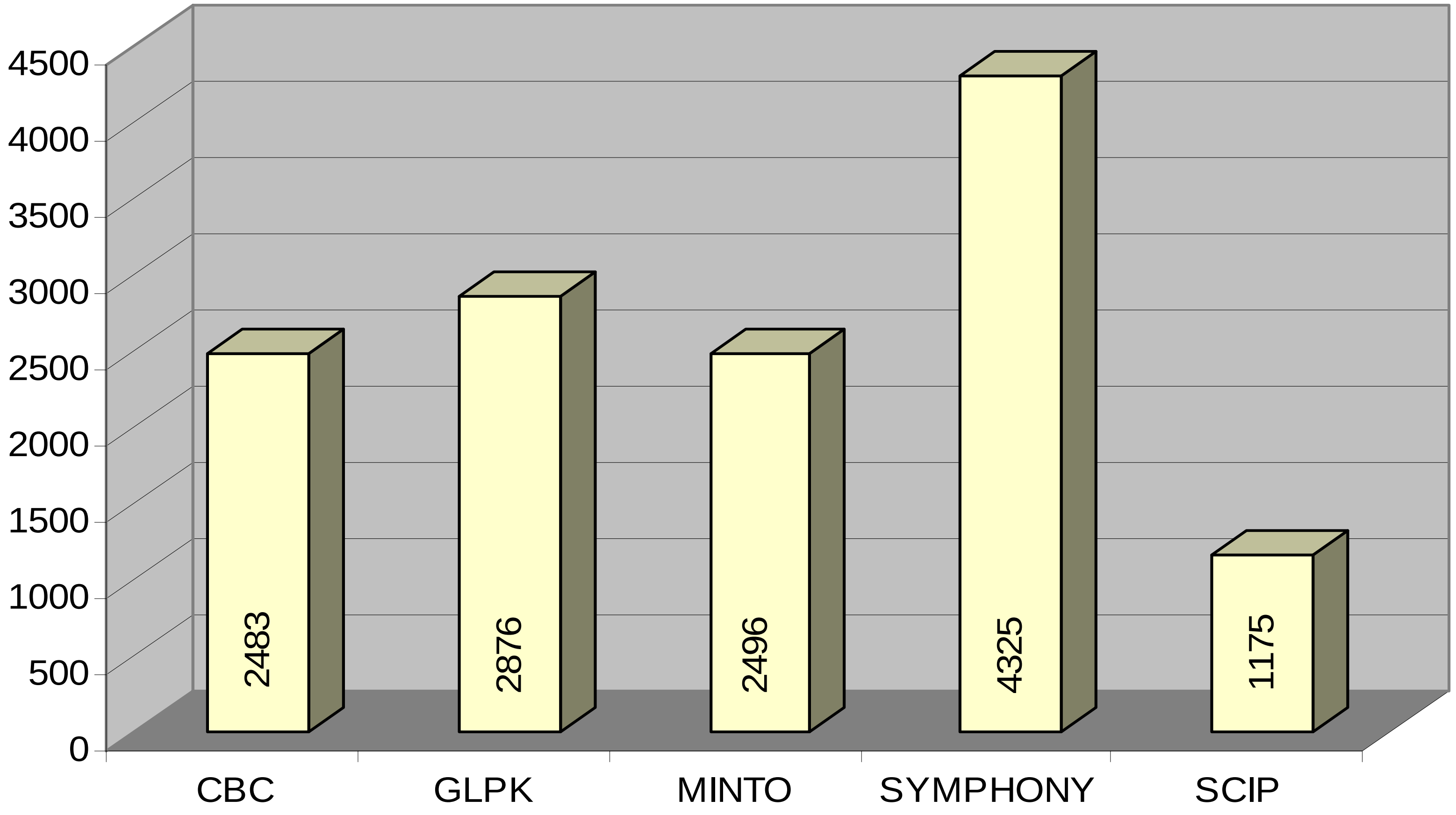
SCIP using CPLEX 10.0 as LP solver:

	CPLEX 10.0	CPLEX 9.1.3	SCIP 0.82
nodes (total)	15 046 000	29 189 000	14 879 000
nodes (geom)	5 278.0	11 703.5	4 704.9
time (total)	29 645.5	41 254.4	33 134.9
time (geom)	119.1	193.9	224.2

SCIP comparable to CPLEX 9.1.3

factor 2 slower than CPLEX 10.0

SCIP using Soplex as LP solver:



geometric mean of solving time (max: 7200 s)

Algebraic Modelling Language

ZIMPL is an algebraic modeling language for mixed integer programming problems.

Modeling languages allow to formulate a mathematical programming problem in a way similar to the original mathematical formulation and automatically generate input for a corresponding solver.

- Problems are represented in a declarative way
- Separation between problem definition and solution process
- Separation between problem structure and data.

Algebraic modeling languages typically operate on contraints consisting of variables indexed by sets connected to parameters by algebraic expressions.

Example

```
# This model solves Sudoku puzzels.
param p := 3;

set L := { 1 .. p*p };
set M := { 1 .. p};

var x[L*L*L] binary;

# file: row col value
set F := { read "sudoku.dat" as "<1n,2n,3n>" comment "#" };

subto dots: forall <i,j> in L*L do sum <k> in L : x[i,j,k] == 1;
subto rows: forall <j,k> in L*L do sum <i> in L : x[i,j,k] == 1;
subto cols: forall <i,k> in L*L do sum <j> in L : x[i,j,k] == 1;
subto sqrs: forall <m,n,k> in M*M*L do
   sum <i,j> in M*M : x[(m-1)*p+i,(n-1)*p+j,k] == 1;

# Fix the fixed values
subto fixed: forall <i,j,k> in F do x[i,j,k] == 1;
```

Extended functions

For $a, b, c \in \mathbb{Z}$, $a \in [-15, 15]$, $b \in [-10, 20]$, $c \in [-20, 10]$,

maximize $5a + 3b + c$ subject to:

$$\begin{aligned} \text{If } & (a \neq b \text{ and } (|a - b| = 3c \text{ or } c - a \geqslant 0)) \\ \text{then } & a + b + c \geqslant 7 \\ \text{else } & a + b \leqslant 1 \end{aligned}$$

can be modeled as an IP, but this very laborious…

Extended functions

For $a, b, c \in \mathbb{Z}$, $a \in [-15, 15], b \in [-10, 20], c \in [-20, 10]$,

maximize $5a + 3b + c$ subject to:

$$\text{If } (a \neq b \text{ and } (|a - b| = 3c \text{ or } c - a \geqslant 0))$$
$$\text{then } a + b + c \geqslant 7$$
$$\text{else } a + b \leqslant 1$$

```
var a integer >= -15 <= 15;
var b integer >= -10 <= 20;
var c integer >= -20 <= 10;
maximize obj: 5 * a + 3 * b + c;
subto c1:
    vif (a!=b and (vabs(a-b) == 3*c or c-a >= 0))
        then a + b + c >= 7
        else a + b <= 1 end;
```

Expansion of extended functions

```
- xp0  + bp2                     <= 0
- b    + a      - xp0  + bm3     <= 0
- xp0  + 25 bp2                  >= 0
- b    + a      - xp0  + 35 bm3  >= 0
- bp2  - bm3    + re4             = 0
- b    + a      + xm6  - xp5      = 0
- xp5  + 25 bp7                  >= 0
+ xm6  + 35 bp7                  <= 35
- xm6  - xp5    + re8             = 0
- xp9  + bp11                    <= 0
- 3 c  + re8    - xp9  + bm12    <= 0
- xp9  + 95 bp11                 >= 0
- 3 c  + re8    - xp9  + 30 bm12 >= 0
+ bp11 + bm12   + re13            = 1
- xp14 + bp16                    <= 0
+ c    - a      - xp14 + bm17    <= 0
- xp14 + 25 bp16                 >= 0
+ c    - a      - xp14 + 35 bm17 >= 0
+ bp16 + bm17                    <= 1
+ re13 - re19                    <= 0
- bm17 - re19                    <= -1
+ re13 - bm17   - re19           >= -1
+ re4  - re20                    >= 0
+ re19 - re20                    >= 0
+ re4  + re19   - re20           <= 1
+ c    + b      + a    - 52 re20 >= -45
+ b    + a             - 34 re20 <= 1
```

IP after presolve:
27 rows, 20 columns, 77 non-zeros.

Optimal solution:

c	2
b	20
a	14
xm1	6
bm3	1
re4	1
xm6	6
re8	6
re13	1
xm15	12
bm17	1
re19	1
re20	1

Zimpl

- is freely available with C source code under the GPL.
- is highly portable (*NIX, Windows, MacOS-X).
- is solver independend.
- is available as a library.
- can be used standalone or linked to a solver.
- is pretty stable.
- has been used in several lectures and industry projects.
- can count correctly.

The Programs

SCIP – Solving Constraint Integer Programs

http://scip.zib.de

SoPlex – Sequential Object-oriented simPlex

http://soplex.zib.de

Zimpl – Zuse Institute Mathematical Programming Language

http://zimpl.zib.de

and friends

perPlex - Rational arithmetic basis verification

http://www.zib.de/koch/perplex

Porta - POlyhedron Representation Transformation Algorithm

http://www.zib.de/Optimization/Software/Porta

MCF - Min Cost Flow http://www.zib.de/Optimization/Software/Mcf

Thank you very much!

Questions?