

Progress in Linear Programming Based Branch-and-Bound Algorithms

Andrew Miller

University of Wisconsin

Objectives

- To discuss what techniques are being used to solve mixed integer programs in practice
- To discuss what can be accomplished with commercial mixed integer optimizers and modeling languages

Outline

- Introduction and linear programming based branch-and-bound
- Branch-and-cut and branch-and-price

Mixed 0-1 Integer Programming

$$\max \quad cx + dy$$

$$Ax + Gy \leq b$$

$$x_j \in \{0,1\}, \quad l_j \leq y_j \leq u_j$$

Why Use MIP ?

- Indivisible commodities
- Binary choices
- Logical relations
- Start up or fixed costs
- Economies of scale
- Combinatorial modeling

Applicability

- Tremendous increase of the use of MIPs in the last decade
- Large-scale MIPs are solvable (provably good solutions) on PCs with commercially available software

Recent Applications

- Transportation planning and operations
- Facility location
- Production scheduling
- Supply-chain management
- Many others

Transportation planning and operations

- Aircraft arrival slot allocation at American. Network optimization used to allocate arrival slots of canceled flights, leading to reduced delays, translates into direct annual operating cost savings of \$5.2M (1991)
- Optimizing airline scheduling of planes and crews. Delta's *Cold Start* solves its fleet assignment with expected savings of \$300M over three years (1994)

Facility location

- Base closing in Germany (US Department of Defense). 0-1 IP used to determine base closures and optimal stationing plan for US troops in Europe after force reductions, annual savings of up to \$58M (1996)

Production scheduling

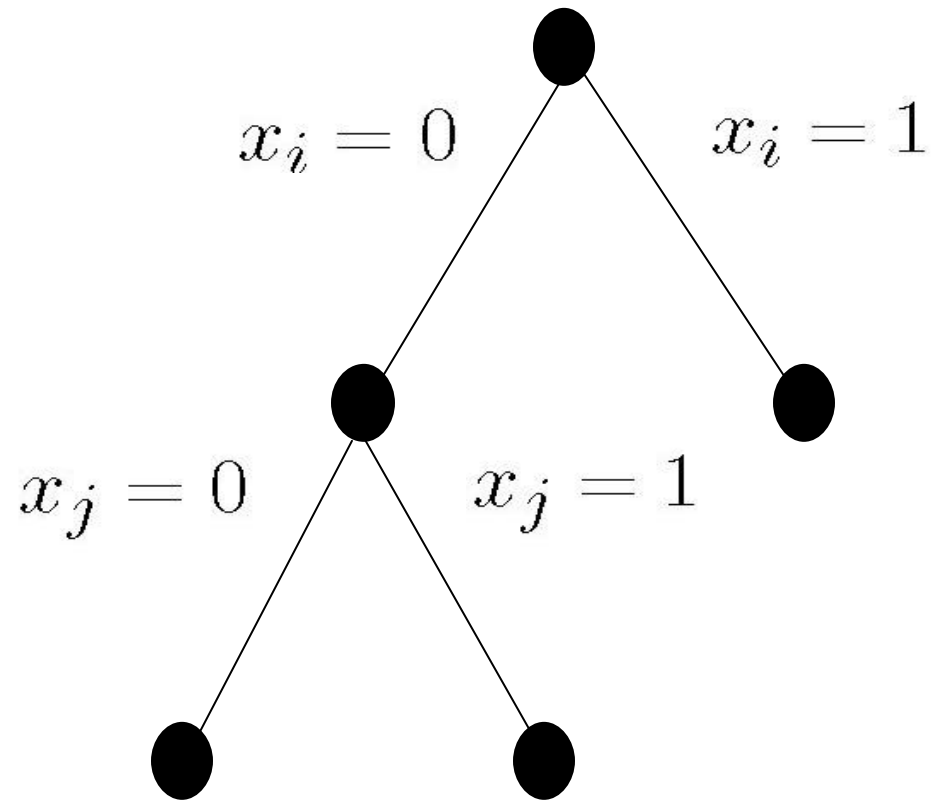
- Fiber-optic cable manufacturing (AT&T).
MIP models select fibers and schedule cable production resulting in more efficient use of fibers and fewer delays in customer orders (1998)

Supply-chain management

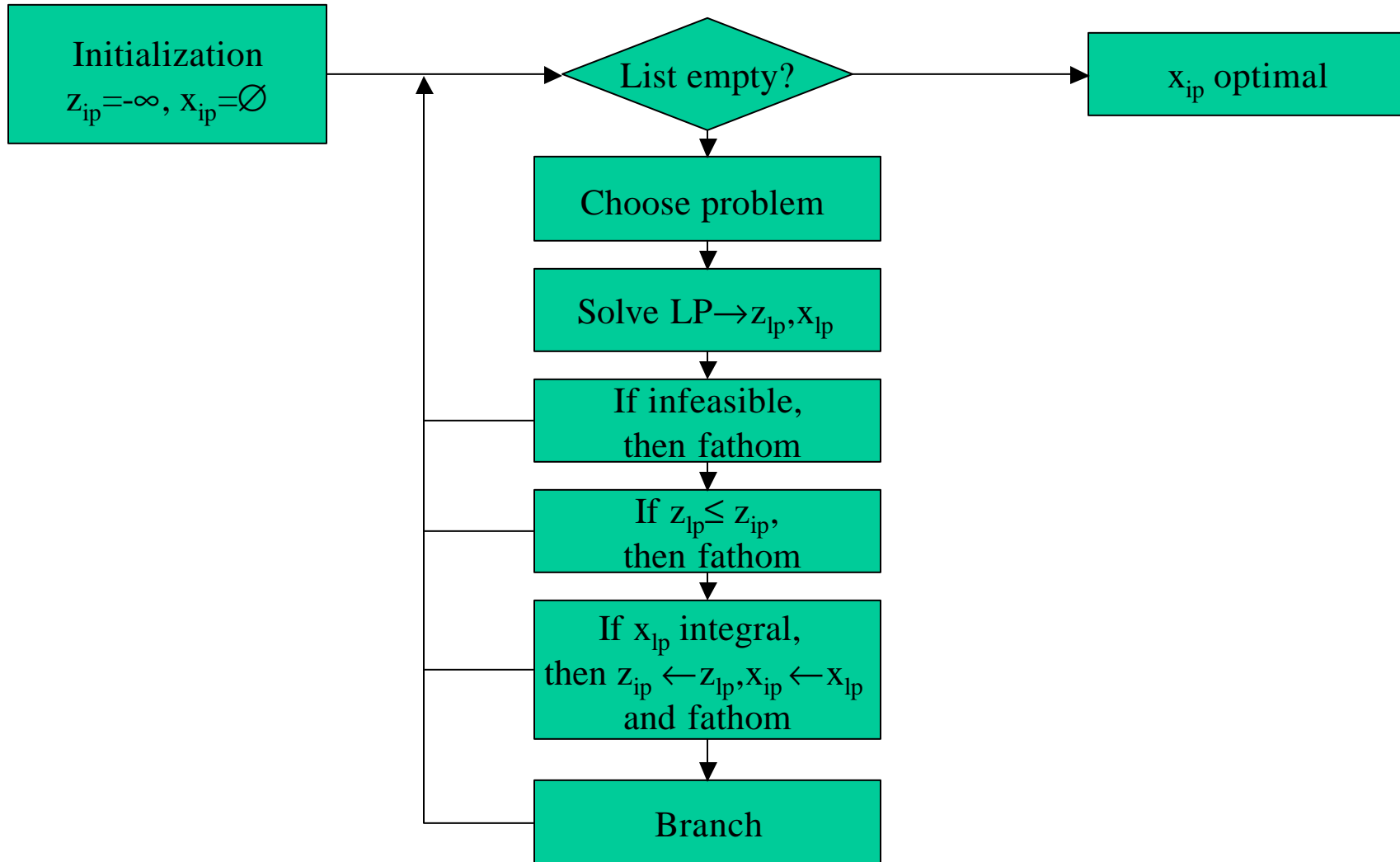
- Global supply chain management (Digital). MIP used to design production, distribution and vendor network so as to minimize cost of production and distribution times, saved \$100M (1995)
- Restructuring the supply chain (Proctor and Gamble). MIP is used to help restructure the supply chain (1997)

LP Based Branch and Bound

- Implicit enumeration tree
- At every node, an LP relaxation is solved
- Upper bounds come from LP solutions; lower bounds come from MIP feasible solutions



LP Based Branch and Bound



How to improve performance?

1. Faster computer
2. Faster linear programming optimizer
3. Smaller z_{lp}
4. Larger z_{ip}
5. Improved branching
6. Smaller linear programs

Formulations

- Most MIPs have many correct formulations, but while one formulation may be easy to solve, another may be very hard to solve
- Failure to solve a MIP may not be the fault of the algorithm, but the result of a “bad” formulation
- Smaller size formulations (number of variables and constraints) may not be better and can be much worse

Uncapacitated Facility Location

x_{ij} : fraction of demand of client i satisfied
from depot j

y_j : 1 if depot j is open
0 otherwise

Uncapacitated Facility Location

$$\sum_{j=1, \dots, n} x_{ij} = 1 \quad i = 1, \dots, m$$

$$\sum_{i=1, \dots, m} x_{ij} \leq my_j \quad j = 1, \dots, n$$

Uncapacitated Facility Location

$$\sum_{j=1, \dots, n} x_{ij} = 1 \quad i = 1, \dots, m$$

$$x_{ij} \leq y_j \quad i = 1, \dots, n, j = 1, \dots, m$$

- *Question: Can we do this automatically ?*

Parallel Machine Scheduling

x_{ij} : 1 if job j is scheduled on machine i

C : makespan

Parallel Machine Scheduling

$$\min C$$

$$\sum_i x_{ij} = 1 \quad \forall j$$

$$\sum_j p_j x_{ij} \leq C \quad \forall i$$

Parallel Machine Scheduling

- Problem: *Symmetry*

Swapping assignments to two machines gives the exact same solution!

- Solution:

Remove symmetry from solution

Parallel Machine Scheduling

$$\min C_m$$

$$\sum_i x_{ij} = 1 \quad \forall j$$

$$\sum_j p_j x_{ij} \leq C_i \quad \forall i$$

$$C_1 \leq C_2 \leq \dots \leq C_m$$

Branching Strategies

- Variable selection
- Node selection

Variable Selection

- Variable dichotomy: $(x_j = \lfloor x_j \rfloor + f_j)$
 $x_j \leq \lfloor x_j \rfloor$
 $x_j \geq \lceil x_j \rceil$
- GOAL: Select a variable that improves the upper bound the most

Prediction Methods

- Estimation methods
- Lower bounding methods

Estimation Methods

Pseudocosts

- P_j^+ Per unit change in objective function value when the variable j is rounded up
- P_j^- Per unit change in objective function value when the variable j is rounded down

Estimates

- $D_j^+ = P_j^+ (1 - f_j)$
- $D_j^- = P_j^- f_j$

Lower Bounding

- L_j^+ Lower bound on the per unit change in objective function value when the variable j is rounded up
- L_j^- Lower bound on the per unit change in objective function value when the variable j is rounded down

Combining Estimation and Lower Bounding

- Estimation provides global information
- Lower bounding provides local information

$$D_j^{new} = \mathbf{b}_1 D_j + \mathbf{b}_2 L_j$$

- Compute parameters a priori
- Compute parameters dynamically

Using Predictors

- How to select branching variable ?

$$\arg \max_j \{D_j^+ + D_j^-\}$$

$$\arg \max_j \{\min(D_j^+, D_j^-\}\}$$

$$\arg \max_j \{\mathbf{a}_1 \max(D_j^+, D_j^-) + \mathbf{a}_2 \min(D_j^+, D_j^-)\}$$

Node Selection

GOAL: (1) Find good feasible solution
(2) Prove optimality

Node Selection Methods

- static
- estimate-based
- two-phase
- backtracking

Static Methods

- best-first (best-bound)
 - advantages
 - minimizes number of evaluated nodes
 - disadvantages
 - high memory requirements
 - high node evaluation times

Static Methods

- depth-first
 - advantages
 - minimizes memory requirements
 - low node evaluation times
 - high chance of finding feasible solutions quickly
 - disadvantages
 - large search trees

Estimate-based Methods

- best projection

$$s = \sum_j \min(f_j, 1 - f_j)$$

$$E = z_{lp} + \left(\frac{z_{best} - z_{lp}^{root}}{s^{root}} \right) s$$

Estimate-based Methods

- best estimate

$$E = z_{lp} - \sum_j \min(|P_j^- f_j|, |P_j^+ (1 - f_j)|)$$

Note: Does not require z_{best}

Two-phase Methods

- Phase I: Find good feasible solution
- Phase II: Prove optimality

Two-phase Methods

- Phase I: Depth-first
- Phase II: Best-first

Two-phase Methods

- Phase I: Best estimate
- Phase II: Maximize Percentage Error

$$PE = 100 \frac{E - z_{best}}{z_{lp} - z_{best}}$$

Backtracking Methods

- Go depth-first as often as possible
- If $z_{1p} \geq E$, then depth-first
- If $z_{1p} \leq E$, then best-first or best-estimate

Preprocessing and Probing

- Preprocessing
 - Detecting infeasibility
 - Detecting redundancy
 - Improving bounds

Detecting Infeasibility

$$z = \min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$l \leq y \leq u$$

$$x \in \{0, 1\}^n, y \in \mathbb{R}^m.$$

Detecting Infeasibility

- If $z > b_i$, then infeasible
- Approximation

$$z \geq z_{LB} > b_i$$

- Simple Approximation

$$- \sum_{j \in B^-} a_j^i + \sum_{j \in C^+} g_j^i l_j - \sum_{j \in C^-} g_j^i u_j > b_i$$

Detecting Infeasibility

$$3x_1 - 4x_2 + 2x_3 \geq 6 \Rightarrow$$

$$-3x_1 + 4x_2 - 2x_3 \leq -6 \Rightarrow$$

$$-3 + 0 - 2 = -5 > -6 \Rightarrow$$

infeasible

Probing

- Tentatively fixing a variable at one of its bounds and examining the consequences
- Three activities
 - Fixing variables
 - Improving coefficients
 - Identifying logical implications

Fixing Variables

$$z_k = \min \sum_{j \in B^+} a_j^i x_j - \sum_{j \in B^-} a_j^i x_j + \sum_{j \in C^+} g_j^i y_j - \sum_{j \in C^-} g_j^i y_j$$

subject to

$$A^i x + G^i y \leq b^i$$

$$x_k = 1$$

$$l \leq y \leq u$$

$$x \in \{0, 1\}^n, y \in \mathbb{R}^m.$$

Fixing Variables

- If $z_k > b_i$, then infeasible
- Conclusion: $x_k \neq 1$ or $x_k = 0$
- Simple Approximation

$$a_k^i - \sum_{j \in B^-} a_j^i + \sum_{j \in C^+} g_j^i l_j - \sum_{j \in C^-} g_j^i u_j > b_i$$

Fixing Variables

Probing $x_2 = 1$

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

$$-3x_1 + 4x_2 - 2x_3 \leq -2 \Rightarrow$$

$$-3 + 4 - 2 = -1 > -2 \Rightarrow$$

infeasible \Rightarrow

$$x_2 = 0$$

Identifying logical implications

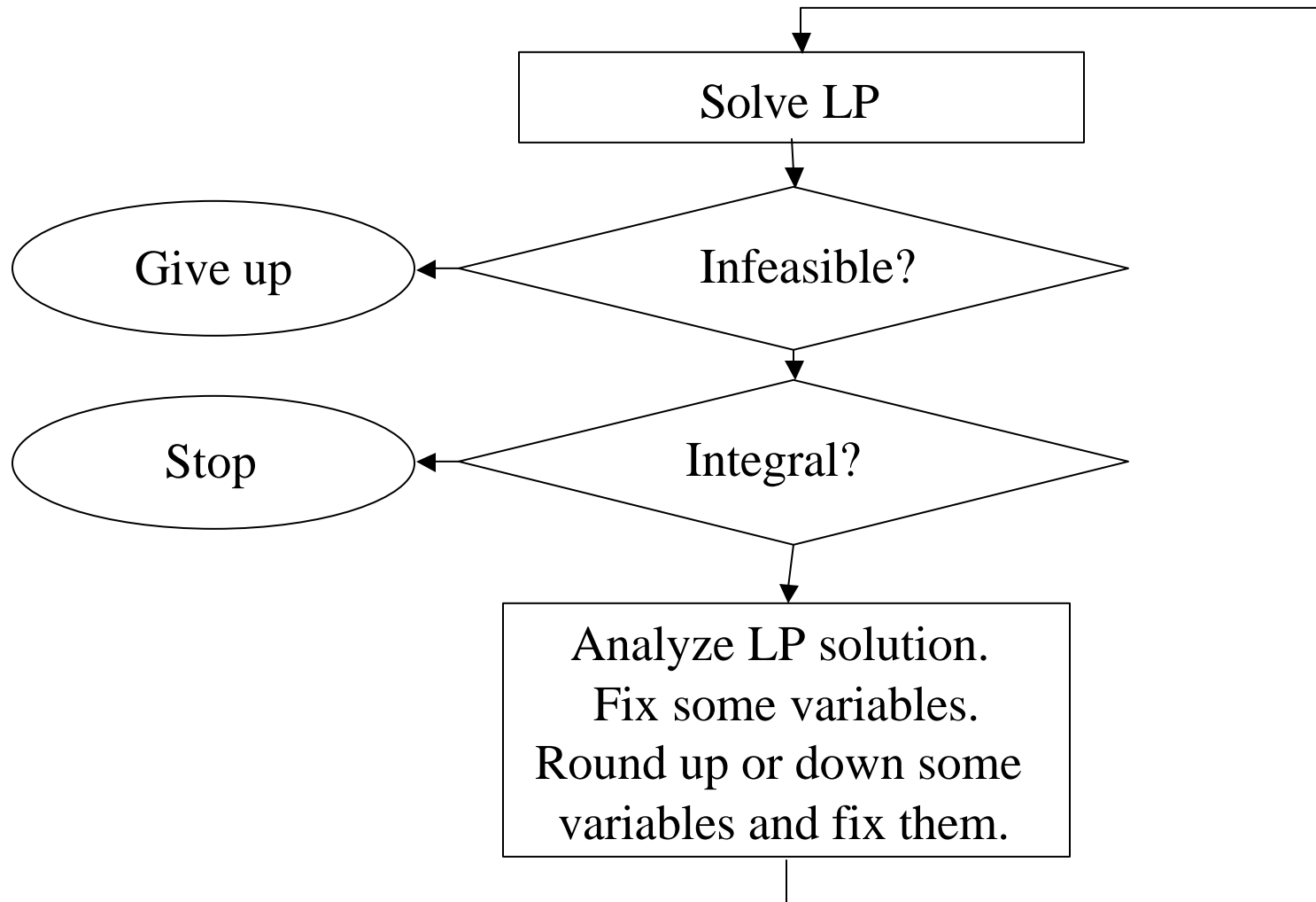
$$x_j = 0, \text{ fixes } x_k = 0 \Rightarrow x_k \leq x_j$$
$$\text{fixes } x_k = 1 \Rightarrow 1 - x_k \leq x_j$$

$$x_j = 1, \text{ fixes } x_k = 0 \Rightarrow x_k \leq 1 - x_j$$
$$\text{fixes } x_k = 1 \Rightarrow 1 - x_k \leq 1 - x_j$$

Identifying logical implications

- $x_i = 1 \Rightarrow y_j = v_j$ implies $y_j \geq l_j + (v_j - l_j)x_i$
- $x_i = 1 \Rightarrow y_j = v_j$ implies $y_j \leq u_j - (u_j - v_j)x_i$
- $x_i = 0 \Rightarrow y_j = v_j$ implies $y_j \geq v_j - (v_j - l_j)x_i$
- $x_i = 0 \Rightarrow y_j = v_j$ implies $y_j \leq v_j + (u_j - v_j)x_i$

Diving Heuristic



Diving Heuristic

- Issues
 - How many variables to fix?
 - How many variables to round and fix?
 - How many iterations?

Local Branching

- Reference solution v

$$S := \{j \mid v_j = 1\}$$

- Local branching constraint

$$\Delta(v) = \sum_{j \in S} (1 - x_j) + \sum_{j \notin S} x_j \leq k$$

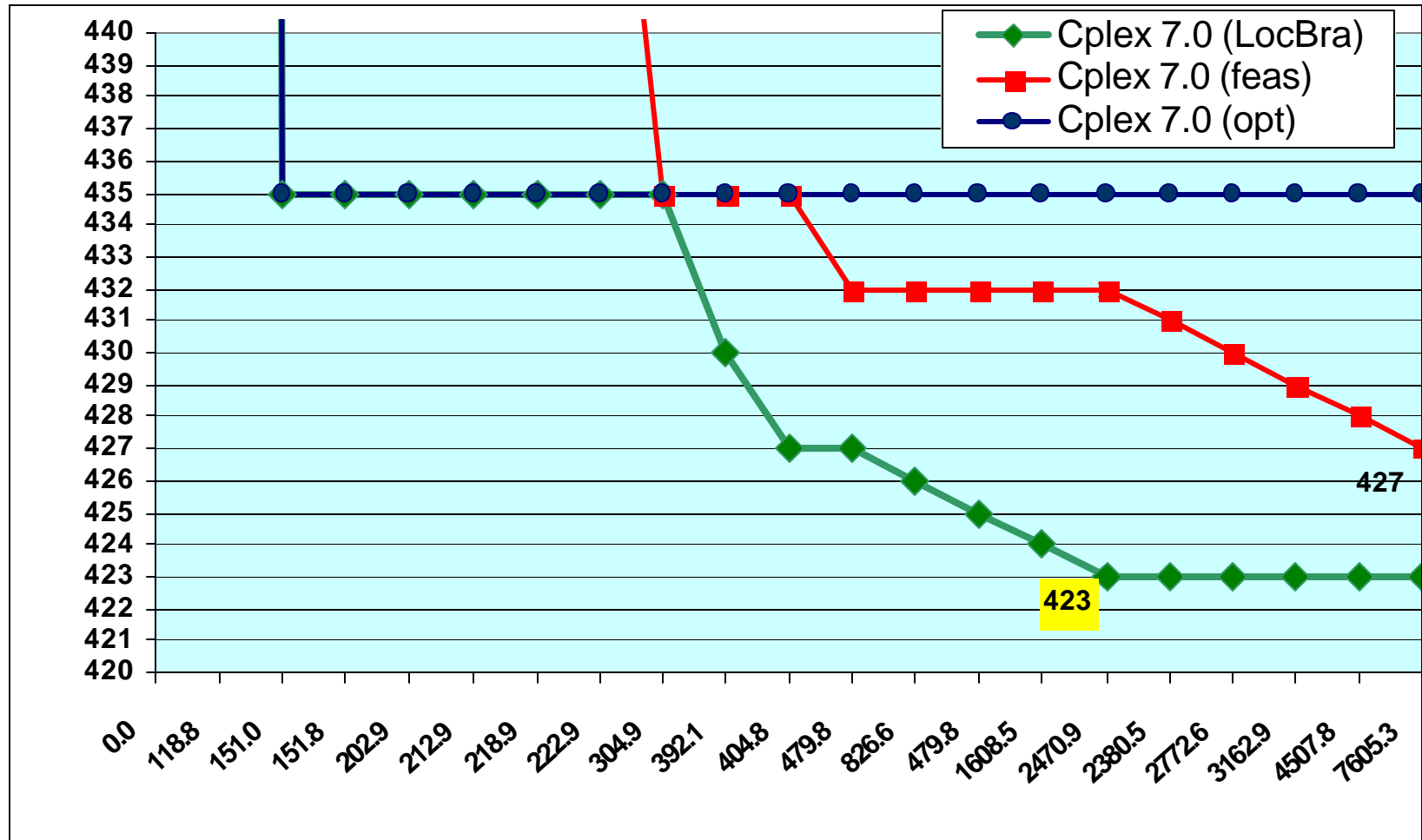
- Left branch

$$\Delta(v) \leq k$$

- Right branch

$$\Delta(v) \geq k+1$$

Local Branching



CPLEX 7.0: MIP emphasis: optimality

- Elapsed b&b time = 151.02 sec. : 435.0
- Final Sol. after 18000.0 sec.s : 435.0

CPLEX 7.0: MIP emphasis: integer feasibility

➤ Elapsed b&b time = 118.85 sec. :	459.0
➤ Elapsed b&b time = 202.98 sec. :	456.0
➤	455.0
➤	454.0
➤ Elapsed b&b time = 222.97 sec. :	453.0
➤ Elapsed b&b time = 304.97 sec. :	435.0
➤ Elapsed b&b time = 479.85 sec. :	432.0
➤ Elapsed b&b time = 2380.52 sec. :	431.0
➤ Elapsed b&b time = 2772.62 sec. :	430.0
➤ Elapsed b&b time = 3162.93 sec. :	429.0
➤ Elapsed b&b time = 4507.88 sec. :	428.0
➤ Elapsed b&b time = 7605.32 sec. :	427.0
➤ Final Sol. after 18000.0 sec.s :	427.0

CPLEX 7.0 & Local Branching

- Local Branch Time = 151.8 sec. : 435.0
- Local Branch Time = 392.1 sec. : 430.0
- Local Branch Time = 404.8 sec. : 427.0
- Local Branch Time = 826.6 sec. : 426.0
- Local Branch Time = 1122.3 sec. : 425.0
- Local Branch Time = 1608.5 sec. : 424.0
- Local Branch Time = 2470.9 sec. : 423.0