Tutorial

Integer Programming for Constraint Programmers

Ambros Gleixner and Stefan Heinz
Zuse Institute Berlin (ZIB)

Chris Beck, Timo Berthold, and Kati Wolter

DFG Research Center MATHEON
Mathematics for key technologies

CP 2011, 14/Sep/2011

ZIB – Zuse Institute Berlin

- Non-university research institute of the state of Berlin (Germany)
- Research Units:
  - Numerical Mathematics
    - Numerical analysis and modeling
    - Visualization and data analysis
  - Discrete Mathematics
    - **Optimization**
    - Scientific Information
  - Computer Science
    - Parallel and Distributed Systems
    - Supercomputing

`http://www.zib.de`


DIE
KLÜGSTE
NACHT
DES
JAHRES
28. Mai 2011
17 BIS 1 UHR

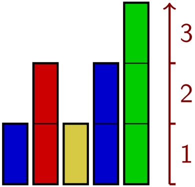

CPAIOR 2012
Nantes, France
May 28 - June 1, 2012
9th International Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming for Combinatorial Optimization Problems

Integer Programming for Constraint Programmers

1. Introduction
2. Linear programming
3. Integer (linear) programming
4. Summary
5. Discussion

Definition

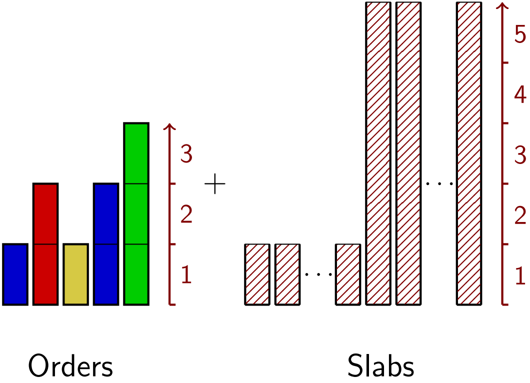
The steel mill slab problem consists of assigning colored, sized orders to slabs of certain different capacities such that the total loss is minimized and at most two different colors are present in each slab.



Orders

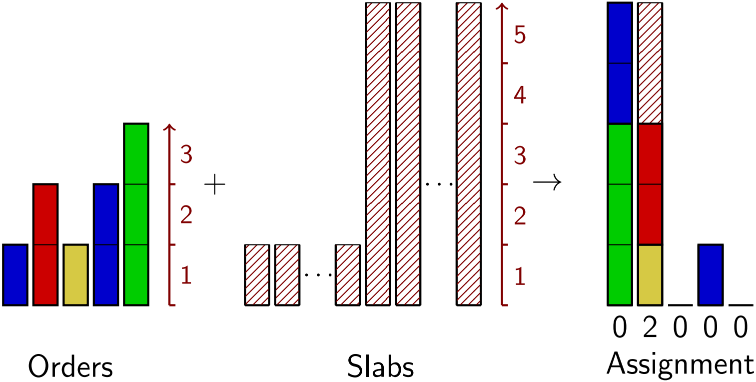
Definition

The steel mill slab problem consists of assigning colored, sized orders to slabs of certain different capacities such that the total loss is minimized and at most two different colors are present in each slab.



Definition

The steel mill slab problem consists of assigning colored, sized orders to slabs of certain different capacities such that the total loss is minimized and at most two different colors are present in each slab.



▷ Problem number 38 of the CSPLib (`http://www.csplib.org/`)

(A) Constraint programming formulation

Given

- ▷ $\mathcal{K}$ set of possible capacities for the slabs
- ▷ $\mathcal{C}$ set of colors
- ▷ $\mathcal{O}$ set of orders, $|\mathcal{O}| = n$
  - s_i size of order i
  - c_i color of order i

Binary variables

- ▷ $y_{ij} = 1$ if order i is assigned to slab j
- ▷ $z_{cj} = 1$ if color c is used in slab j

Observation

- ▷ We need at most n slabs
- ▷ Let $\mathcal{S}$ be the set of slabs

- Array storing the leftover depending on the load

$$\mathcal{L}[i] = \text{argmin}\{k - i \mid k \in \mathcal{K} \text{ and } k \geq i\}$$

  for $i = 0, \ldots, \mathcal{K}_{\max}$
- $\mathcal{K}_{\max} := \max\{k \mid k \in \mathcal{K}\}$

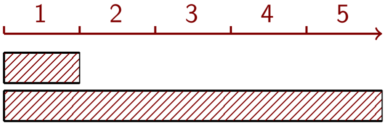
Leftover array

- Array storing the leftover depending on the load

$$\mathcal{L}[i] = \text{argmin}\{k - i \mid k \in \mathcal{K} \text{ and } k \geq i\}$$

  for $i = 0, \ldots, \mathcal{K}_{\max}$
- $\mathcal{K}_{\max} := \max\{k \mid k \in \mathcal{K}\}$

Example



$$\mathcal{L} = \boxed{0}\,\boxed{}\,\boxed{}\,\boxed{}\,\boxed{}\,\boxed{}$$

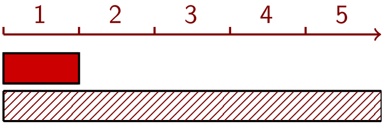
Leftover array

- Array storing the leftover depending on the load

$$\mathcal{L}[i] = \operatorname{argmin}\{k - i \mid k \in \mathcal{K} \text{ and } k \geq i\}$$

  for $i = 0, \ldots, \mathcal{K}_{\max}$
- $\mathcal{K}_{\max} := \max\{k \mid k \in \mathcal{K}\}$

Example



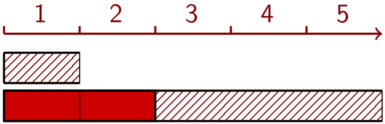
$$\mathcal{L} = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & & & & \\ \hline \end{array}$$

- Array storing the leftover depending on the load

$$\mathcal{L}[i] = \text{argmin}\{k - i \mid k \in \mathcal{K} \text{ and } k \geq i\}$$

  for $i = 0, \ldots, \mathcal{K}_{\max}$
- $\mathcal{K}_{\max} := \max\{k \mid k \in \mathcal{K}\}$

Example



$\mathcal{L} =$

0	0	3			

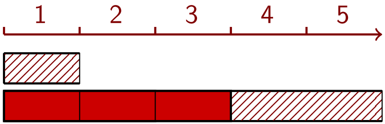
Leftover array

- Array storing the leftover depending on the load

$$\mathcal{L}[i] = \text{argmin}\{k - i \mid k \in \mathcal{K} \text{ and } k \geq i\}$$

  for $i = 0, \ldots, \mathcal{K}_{\max}$
- $\mathcal{K}_{\max} := \max\{k \mid k \in \mathcal{K}\}$

Example



$\mathcal{L} =$

0	0	3	2		

Leftover array

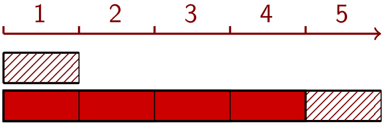
- Array storing the leftover depending on the load

$$\mathcal{L}[i] = \text{argmin}\{k - i \mid k \in \mathcal{K} \text{ and } k \geq i\}$$

  for $i = 0, \ldots, \mathcal{K}_{\max}$

- $\mathcal{K}_{\max} := \max\{k \mid k \in \mathcal{K}\}$

Example



$$\mathcal{L} = \begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 3 & 2 & 1 & \\ \hline\end{array}$$

Leftover array

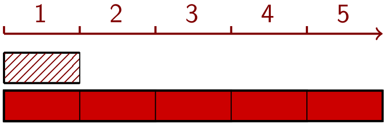
- Array storing the leftover depending on the load

$$\mathcal{L}[i] = \operatorname{argmin}\{k - i \mid k \in \mathcal{K} \text{ and } k \geq i\}$$

  for $i = 0, \ldots, \mathcal{K}_{\max}$

- $\mathcal{K}_{\max} := \max\{k \mid k \in \mathcal{K}\}$

Example



$$\mathcal{L} = \begin{array}{|c|c|c|c|c|c|} \hline 0 & 0 & 3 & 2 & 1 & 0 \\ \hline \end{array}$$

(A) Constraint programming formulation

$$
\begin{array}{lll}
\min & \displaystyle\sum_{j\in\mathcal{S}} \mathcal{L}\left[\sum_{i\in\mathcal{O}} s_i\, y_{ij}\right] & \\
\text{subject to} & \displaystyle\sum_{j\in\mathcal{S}} y_{ij} = 1 & \forall i \in \mathcal{O} \\
& \displaystyle\sum_{i\in\mathcal{O}} s_i\, y_{ij} \leq \mathcal{K}_{\max} & \forall j \in \mathcal{S} \\
& y_{ij} \leq z_{c_i j} & \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} \\
& \displaystyle\sum_{c\in\mathcal{C}} z_{cj} \leq 2 & \forall j \in \mathcal{S} \\
& y_{ij}, z_{cj} \in \{0,1\} & \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S}
\end{array}
$$

(A) Constraint programming formulation

$$
\begin{array}{llll}
\min & \displaystyle\sum_{j\in\mathcal{S}} \mathcal{L}\left[\sum_{i\in\mathcal{O}} s_i\, y_{ij}\right] & & \\
\text{subject to} & \displaystyle\sum_{j\in\mathcal{S}} y_{ij} = 1 & \forall i \in \mathcal{O} & \text{Assignment} \\
& \displaystyle\sum_{i\in\mathcal{O}} s_i\, y_{ij} \le \mathcal{K}_{\max} & \forall j \in \mathcal{S} & \\
& y_{ij} \le z_{c_i j} & \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} & \\
& \displaystyle\sum_{c\in\mathcal{C}} z_{cj} \le 2 & \forall j \in \mathcal{S} & \\
& y_{ij}, z_{cj} \in \{0,1\} & \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S} &
\end{array}
$$

(A) Constraint programming formulation

$$
\begin{array}{llll}
\min & \sum_{j \in \mathcal{S}} \mathcal{L}\left[\sum_{i \in \mathcal{O}} s_i \, y_{ij}\right] & & \\
\text{subject to} & \sum_{j \in \mathcal{S}} y_{ij} = 1 & \forall i \in \mathcal{O} & \\
& \sum_{i \in \mathcal{O}} s_i \, y_{ij} \le \mathcal{K}_{\max} & \forall j \in \mathcal{S} & \text{Capacity} \\
& y_{ij} \le z_{c_i j} & \forall i \in \mathcal{O} \; \forall j \in \mathcal{S} & \\
& \sum_{c \in \mathcal{C}} z_{cj} \le 2 & \forall j \in \mathcal{S} & \\
& y_{ij}, z_{cj} \in \{0, 1\} & \forall i \in \mathcal{O} \; \forall c \in \mathcal{C} \; \forall j \in \mathcal{S} &
\end{array}
$$

(A) Constraint programming formulation

$$
\begin{aligned}
\min \quad & \sum_{j \in \mathcal{S}} \mathcal{L}\left[\sum_{i \in \mathcal{O}} s_i \, y_{ij}\right] && \\
\text{subject to} \quad & \sum_{j \in \mathcal{S}} y_{ij} = 1 && \forall i \in \mathcal{O} \\
& \sum_{i \in \mathcal{O}} s_i \, y_{ij} \leq \mathcal{K}_{\max} && \forall j \in \mathcal{S} \\
& y_{ij} \leq z_{c_i j} && \forall i \in \mathcal{O} \; \forall j \in \mathcal{S} \quad \text{Coloring} \\
& \sum_{c \in \mathcal{C}} z_{cj} \leq 2 && \forall j \in \mathcal{S} \\
& y_{ij}, z_{cj} \in \{0, 1\} && \forall i \in \mathcal{O} \; \forall c \in \mathcal{C} \; \forall j \in \mathcal{S}
\end{aligned}
$$

(A) Constraint programming formulation

$$\min \quad \sum_{j \in \mathcal{S}} \mathcal{L}\left[\sum_{i \in \mathcal{O}} s_i \, y_{ij}\right] \qquad \text{Leftover}$$

$$\begin{aligned}
\text{subject to} \quad & \sum_{j \in \mathcal{S}} y_{ij} = 1 && \forall i \in \mathcal{O} \\
& \sum_{i \in \mathcal{O}} s_i \, y_{ij} \leq \mathcal{K}_{\max} && \forall j \in \mathcal{S} \\
& y_{ij} \leq z_{c_i j} && \forall i \in \mathcal{O} \; \forall j \in \mathcal{S} \\
& \sum_{c \in \mathcal{C}} z_{cj} \leq 2 && \forall j \in \mathcal{S} \\
& y_{ij}, z_{cj} \in \{0, 1\} && \forall i \in \mathcal{O} \; \forall c \in \mathcal{C} \; \forall j \in \mathcal{S}
\end{aligned}$$

- ELEMENT constraint

(An) Integer programming formulation

Given

- $\mathcal{K}$ set of possible capacities for the slabs
- $\mathcal{C}$ set of colors
- $\mathcal{O}$ set of orders, $|\mathcal{O}| = n$
  - s_i size of order i
  - c_i color of order i

Binary variables

- $x_{kj} = 1$ if capacity k is assigned to slab j
- $y_{ij} = 1$ if order i is assigned to slab j
- $z_{cj} = 1$ if color c is used in slab j

Observation

- We need at most n slabs
- Let $\mathcal{S}$ be the set of slabs

(An) Integer programming formulation

$$
\begin{aligned}
\text{min} \quad & \sum_{j \in \mathcal{S}} \sum_{k \in \mathcal{K}} k\, x_{kj} - \sum_{i \in \mathcal{O}} s_i \\
\text{subject to} \quad & \sum_{k \in \mathcal{K}} x_{kj} = 1 && \forall j \in \mathcal{S} \\
& \sum_{j \in \mathcal{S}} y_{ij} = 1 && \forall i \in \mathcal{O} \\
& \sum_{i \in \mathcal{O}} s_i\, y_{ij} \leq \sum_{k \in \mathcal{K}} k\, x_{kj} && \forall j \in \mathcal{S} \\
& y_{ij} \leq z_{c_i j} && \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} \\
& \sum_{c \in \mathcal{C}} z_{cj} \leq 2 && \forall j \in \mathcal{S} \\
& x_{kj}, y_{ij}, z_{cj} \in \{0, 1\} && \forall k \in \mathcal{K}\ \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S}
\end{aligned}
$$

(An) Integer programming formulation

$$
\begin{array}{llll}
\min & \displaystyle\sum_{j\in\mathcal{S}}\sum_{k\in\mathcal{K}} k\, x_{kj} - \sum_{i\in\mathcal{O}} s_i & & \\
\text{subject to} & \displaystyle\sum_{k\in\mathcal{K}} x_{kj} = 1 & \forall j \in \mathcal{S} & \text{Assignment} \\
& \displaystyle\sum_{j\in\mathcal{S}} y_{ij} = 1 & \forall i \in \mathcal{O} & \\
& \displaystyle\sum_{i\in\mathcal{O}} s_i\, y_{ij} \le \sum_{k\in\mathcal{K}} k\, x_{kj} & \forall j \in \mathcal{S} & \\
& y_{ij} \le z_{c_i j} & \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} & \\
& \displaystyle\sum_{c\in\mathcal{C}} z_{cj} \le 2 & \forall j \in \mathcal{S} & \\
& x_{kj}, y_{ij}, z_{cj} \in \{0,1\} & \forall k \in \mathcal{K}\ \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S} &
\end{array}
$$

(An) Integer programming formulation

$$
\begin{aligned}
\text{min} \quad & \sum_{j\in\mathcal{S}} \sum_{k\in\mathcal{K}} k\, x_{kj} - \sum_{i\in\mathcal{O}} s_i \\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} x_{kj} = 1 && \forall j \in \mathcal{S} \\
& \sum_{j\in\mathcal{S}} y_{ij} = 1 && \forall i \in \mathcal{O} \\
& \sum_{i\in\mathcal{O}} s_i\, y_{ij} \le \sum_{k\in\mathcal{K}} k\, x_{kj} && \forall j \in \mathcal{S} && \text{Capacity} \\
& y_{ij} \le z_{c_i j} && \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} \\
& \sum_{c\in\mathcal{C}} z_{cj} \le 2 && \forall j \in \mathcal{S} \\
& x_{kj}, y_{ij}, z_{cj} \in \{0,1\} && \forall k \in \mathcal{K}\ \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S}
\end{aligned}
$$

(An) Integer programming formulation

$$
\begin{aligned}
\text{min} \quad & \sum_{j\in\mathcal{S}} \sum_{k\in\mathcal{K}} k\, x_{kj} - \sum_{i\in\mathcal{O}} s_i \\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} x_{kj} = 1 && \forall j \in \mathcal{S} \\
& \sum_{j\in\mathcal{S}} y_{ij} = 1 && \forall i \in \mathcal{O} \\
& \sum_{i\in\mathcal{O}} s_i\, y_{ij} \le \sum_{k\in\mathcal{K}} k\, x_{kj} && \forall j \in \mathcal{S} \\
& y_{ij} \le z_{c_i j} && \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} && \text{Coloring} \\
& \sum_{c\in\mathcal{C}} z_{cj} \le 2 && \forall j \in \mathcal{S} \\
& x_{kj}, y_{ij}, z_{cj} \in \{0,1\} && \forall k \in \mathcal{K}\ \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S}
\end{aligned}
$$

(An) Integer programming formulation

$$\min \sum_{j \in \mathcal{S}} \sum_{k \in \mathcal{K}} k\, x_{kj} - \sum_{i \in \mathcal{O}} s_i$$

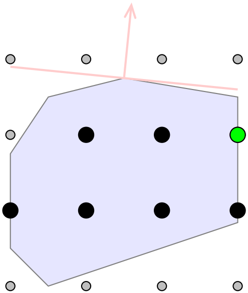
Leftover

$$
\begin{array}{lll}
\text{subject to} & \displaystyle\sum_{k \in \mathcal{K}} x_{kj} = 1 & \forall j \in \mathcal{S} \\
& \displaystyle\sum_{j \in \mathcal{S}} y_{ij} = 1 & \forall i \in \mathcal{O} \\
& \displaystyle\sum_{i \in \mathcal{O}} s_i\, y_{ij} \leq \sum_{k \in \mathcal{K}} k\, x_{kj} & \forall j \in \mathcal{S} \\
& y_{ij} \leq z_{c_i j} & \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} \\
& \displaystyle\sum_{c \in \mathcal{C}} z_{cj} \leq 2 & \forall j \in \mathcal{S} \\
& x_{kj}, y_{ij}, z_{cj} \in \{0, 1\} & \forall k \in \mathcal{K}\ \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S}
\end{array}
$$

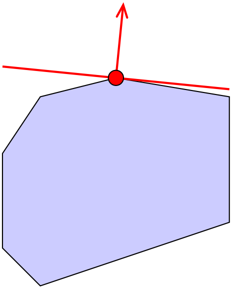
```
###################
#
# data parsing
#
###################

# number of capacities
param ncapacity := read DATAFILE as "1n" use 1;
do print "ncapacity = ", ncapacity;

# number of colors
param ncolors := read  DATAFILE as "1n" skip 1 use 1;
do print "ncolors = ", ncolors;

# number of orderns
param norders := read  DATAFILE as "1n" skip 2 use 1;
do print "norders = ", norders;

# get the capacity array
set tmp := {read DATAFILE as "<n+>" use 1};
set capacities := if card(tmp) == ncapacity then tmp union {0} else tmp union {0} \ {ncapacity} end ;
do check ncapacity == card(capacities)-1;
do print capacities;

# index set for orders
set I := {1..norders};

# index set for colors
set C := {1..ncolors};

# get orders
set orders[<i> in I] := {read DATAFILE as "<1n,2n>" skip 2+i use 1};
#do forall <i> in I do print orders[i];

###################
#
# decision variables
#
###################

# slab variables which capacities is assigned to which slab
var x[I * capacities] binary;

# which order is assigned to which slab
var y[I * I] binary;

# which color is used by which slab
var z[C * I];

###################
#
# objective function
#
###################

minimize leftover:
         sum <s,c> in I * capacities : c * x[s,c] - sum <o,s> in I * I : ord(orders[o],1,1) * y[o,s];

###################
#
# constraints
#
###################

# each slab gets exactly one capacities
subto oneCapacity:
        forall <s> in I : sum <c> in capacities: x[s,c] == 1;

# each order is assigned to exactly one slab
subto oneSlab:
        forall <o> in I : sum <s> in I: y[o,s] == 1;

# each slab is not over loaded
subto Capacity:
        forall <s> in I : sum <c> in capacities: c * x[s,c] - sum <o> in I: ord(orders[o],1,1) * y[o,s] >= 0;

# color linking constraints
subto linking:
        forall <o,s> in I * I : y[o,s] - z[ord(orders[o],1,2),s] <= 0;

# color linking constraints
subto Color:
        forall <s> in I : sum <c> in C : z[c,s] <= 2;
```

Integer program

Linear program relaxation



$\rightarrow$



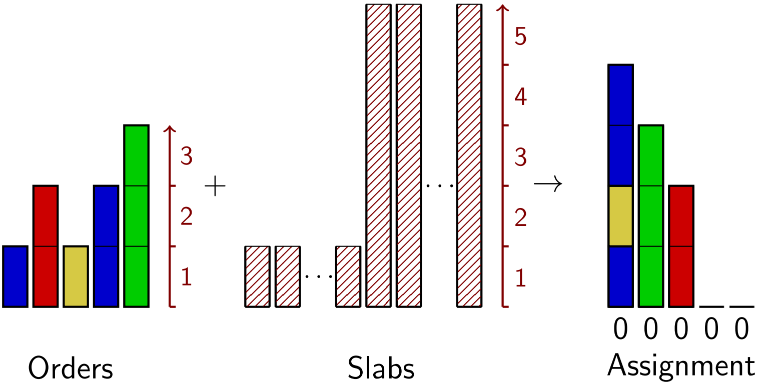
$$\min\{IP\} \geq \min\{LP\}$$

$$\max\{IP\} \leq \max\{LP\}$$

▷ Omit integrality condition

$$
\begin{aligned}
\min \quad & \sum_{j\in\mathcal{S}} \sum_{k\in\mathcal{K}} k\, x_{kj} - \sum_{i\in\mathcal{O}} s_i \\
\text{subject to} \quad & \sum_{k\in\mathcal{K}} x_{kj} = 1 && \forall j \in \mathcal{S} \\
& \sum_{j\in\mathcal{S}} y_{ij} = 1 && \forall i \in \mathcal{O} \\
& \sum_{i\in\mathcal{O}} s_i\, y_{ij} \leq \sum_{k\in\mathcal{K}} k\, x_{kj} && \forall j \in \mathcal{S} \\
& y_{ij} \leq z_{c_i j} && \forall i \in \mathcal{O}\ \forall j \in \mathcal{S} \\
& \sum_{c\in\mathcal{C}} z_{cj} \leq 2 && \forall j \in \mathcal{S} \\
& x_{kj}, y_{ij}, z_{cj} \in [0,1] && \forall k \in \mathcal{K}\ \forall i \in \mathcal{O}\ \forall c \in \mathcal{C}\ \forall j \in \mathcal{S}
\end{aligned}
$$

Root node solution



Root node solution

Capacities

$$x_{51} = 0.8$$
$$x_{01} = 0.2$$
$$x_{52} = 0.6$$
$$x_{02} = 0.4$$
$$x_{53} = 0.4$$
$$x_{03} = 0.6$$
$$x_{04} = 1.0$$
$$x_{06} = 1.0$$

Assignments

$$y_{11} = 1.0$$
$$y_{23} = 1.0$$
$$y_{31} = 1.0$$
$$y_{41} = 1.0$$
$$y_{52} = 1.0$$

Colors

$$z_{11} = 1.0$$
$$z_{23} = 1.0$$
$$z_{31} = 1.0$$
$$z_{32} = 1.0$$
$$z_{33} = 1.0$$
$$z_{34} = 1.0$$
$$z_{35} = 1.0$$
$$z_{42} = 1.0$$

- Remaining decission variables are zero

Capacities

$$x_{51} = 0.8$$
$$x_{01} = 0.2$$
$$x_{52} = 0.6$$
$$x_{02} = 0.4$$
$$x_{53} = 0.4$$
$$x_{03} = 0.6$$
$$x_{04} = 1.0$$
$$x_{06} = 1.0$$

Assignments

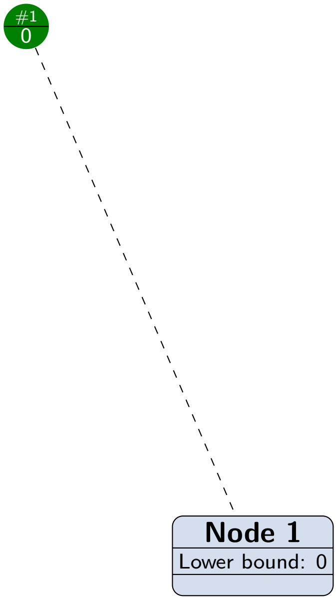
$$y_{11} = 1.0$$
$$y_{23} = 1.0$$
$$y_{31} = 1.0$$
$$y_{41} = 1.0$$
$$y_{52} = 1.0$$

Colors

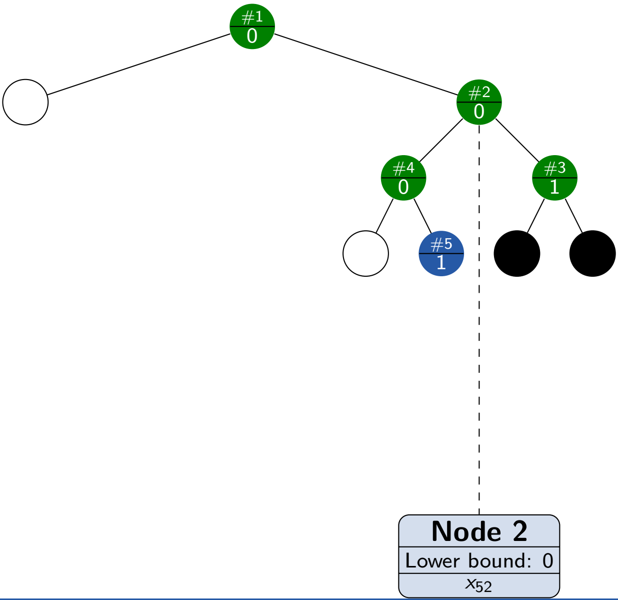
$$z_{11} = 1.0$$
$$z_{23} = 1.0$$
$$z_{31} = 1.0$$
$$z_{32} = 1.0$$
$$z_{33} = 1.0$$
$$z_{34} = 1.0$$
$$z_{35} = 1.0$$
$$z_{42} = 1.0$$

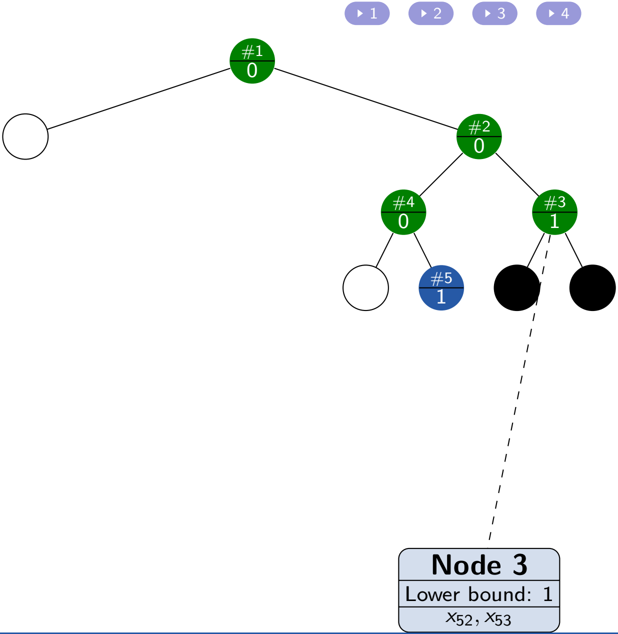
- Remaining decission variables are zero
- Observation: Independently of the problem instance the root LP value for this model is always zero.

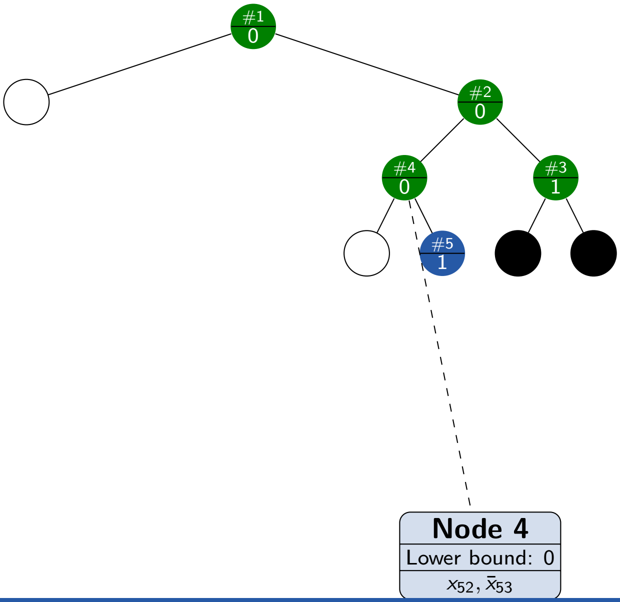
node	left	depth	frac	curdualbound	dualbound	primalbound
1	0	0	6	0.000000e+00	0.000000e+00	--


#1
0
Node 1
Lower bound: 0

▸ 1 ▸ 2 ▸ 3 ▸ 4

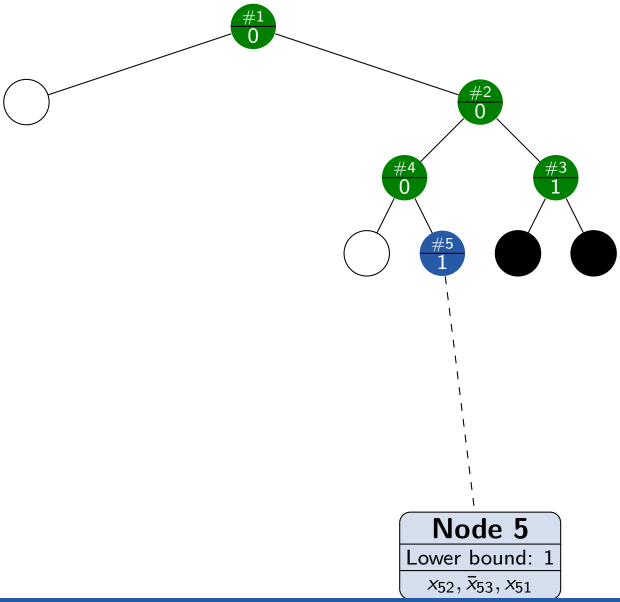



1
2
3
4
#1
0
#2
0
#4
0
#3
1
#5
1
Node 3
Lower bound: 1
x_{52}, x_{53}

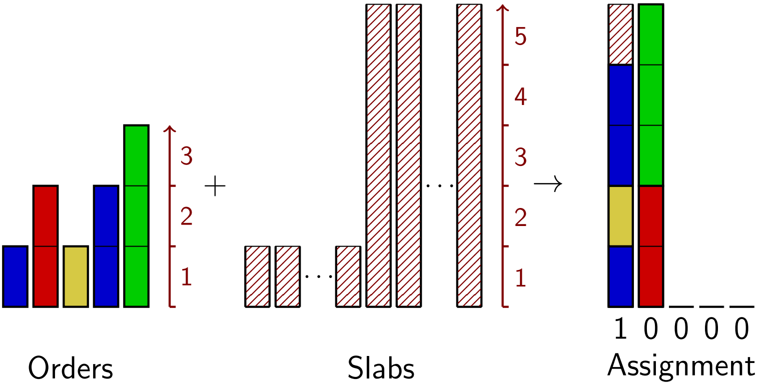

#1
0
#2
0
#4
0
#3
1
#5
1
Node 4
Lower bound: 0
$x_{52}, \bar{x}_{53}$

Search tree

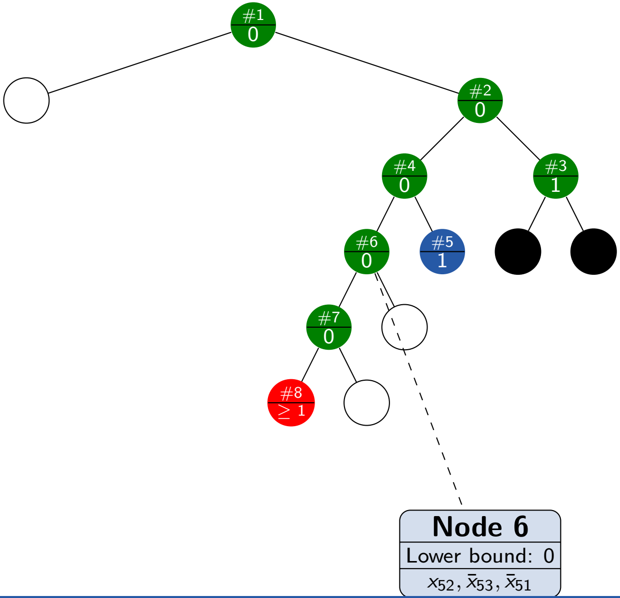
▸ 1 ▸ 2 ▸ 3 ▸ 4



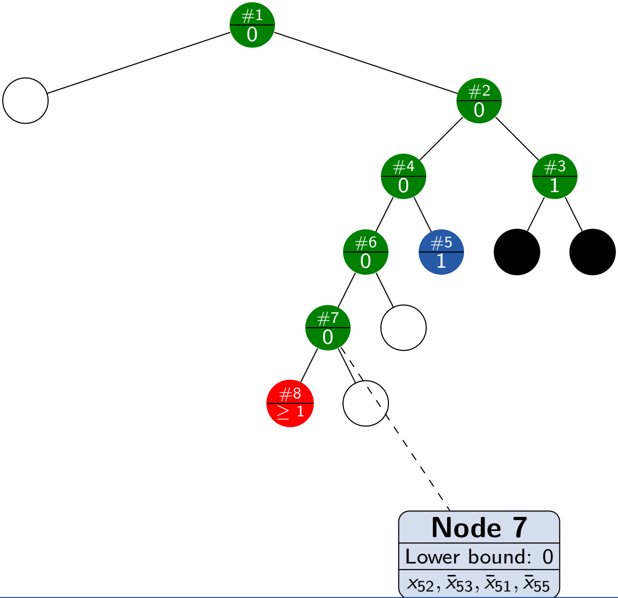
	node	left	depth	frac	curdualbound	dualbound	primalbound
	1	0	0	6	0.000000e+00	0.000000e+00	--
	1	2	0	6	0.000000e+00	0.000000e+00	--
	2	3	1	4	0.000000e+00	0.000000e+00	--
	3	4	2	6	1.000000e+00	0.000000e+00	--
	4	5	2	2	0.000000e+00	0.000000e+00	--
*	5	2	3	-	1.000000e+00	0.000000e+00	1.000000e+00


3
2
1
+
5
4
3
2
1
→
1 0 0 0 0
Orders
Slabs
Assignment

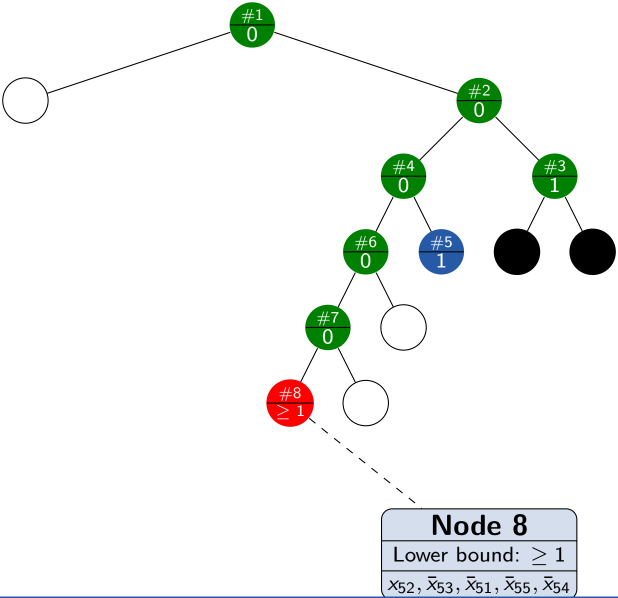
▸ 1 ▸ 2 ▸ 3 ▸ 4



▸ 1 ▸ 2 ▸ 3 ▸ 4

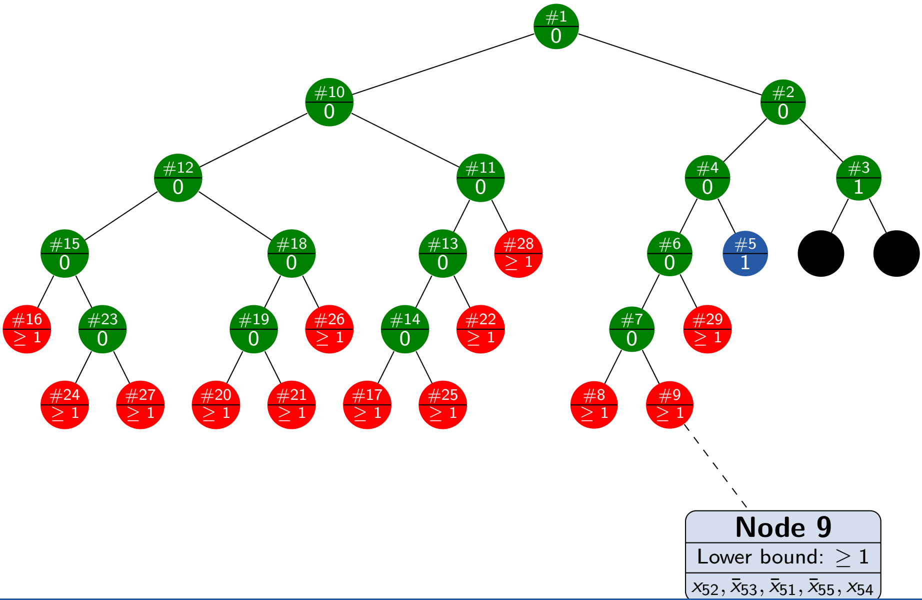

#1
0
#2
0
#4
0
#3
1
#6
0
#5
1
#7
0
#8
≥ 1
Node 7
Lower bound: 0
$x_{52}, \bar{x}_{53}, \bar{x}_{51}, \bar{x}_{55}$

▸ 1 ▸ 2 ▸ 3 ▸ 4

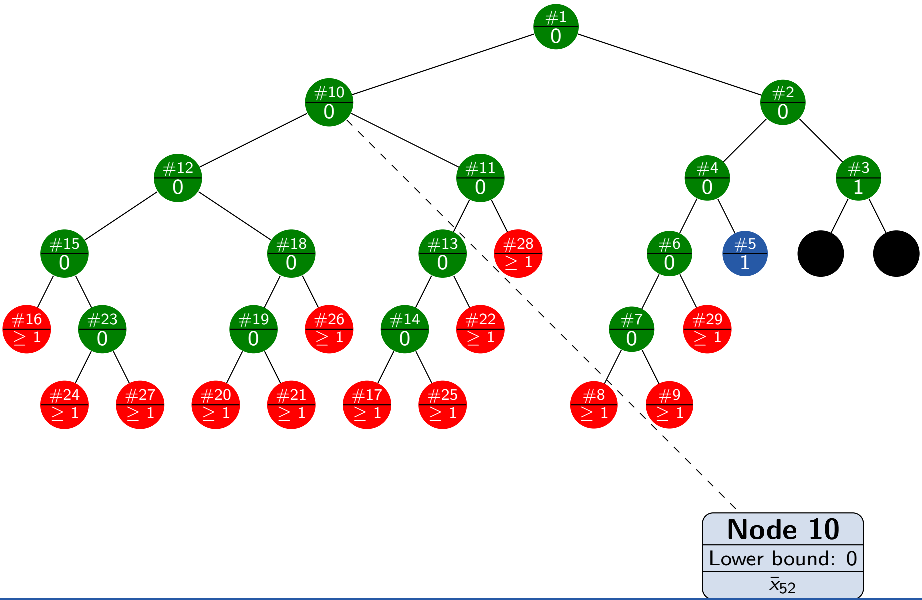

#1
0
#2
0
#4
0
#3
1
#6
0
#5
1
#7
0
#8
≥ 1
Node 8
Lower bound: ≥ 1
$x_{52}, \bar{x}_{53}, \bar{x}_{51}, \bar{x}_{55}, \bar{x}_{54}$

	node	left	depth	frac	curdualbound	dualbound	primalbound
	1	0	0	6	0.000000e+00	0.000000e+00	--
	1	2	0	6	0.000000e+00	0.000000e+00	--
	2	3	1	4	0.000000e+00	0.000000e+00	--
	3	4	2	6	1.000000e+00	0.000000e+00	--
	4	5	2	2	0.000000e+00	0.000000e+00	--
*	5	2	3	-	1.000000e+00	0.000000e+00	1.000000e+00
	6	3	3	2	0.000000e+00	0.000000e+00	1.000000e+00
	7	4	4	2	0.000000e+00	0.000000e+00	1.000000e+00
	8	3	5	-	--	0.000000e+00	1.000000e+00

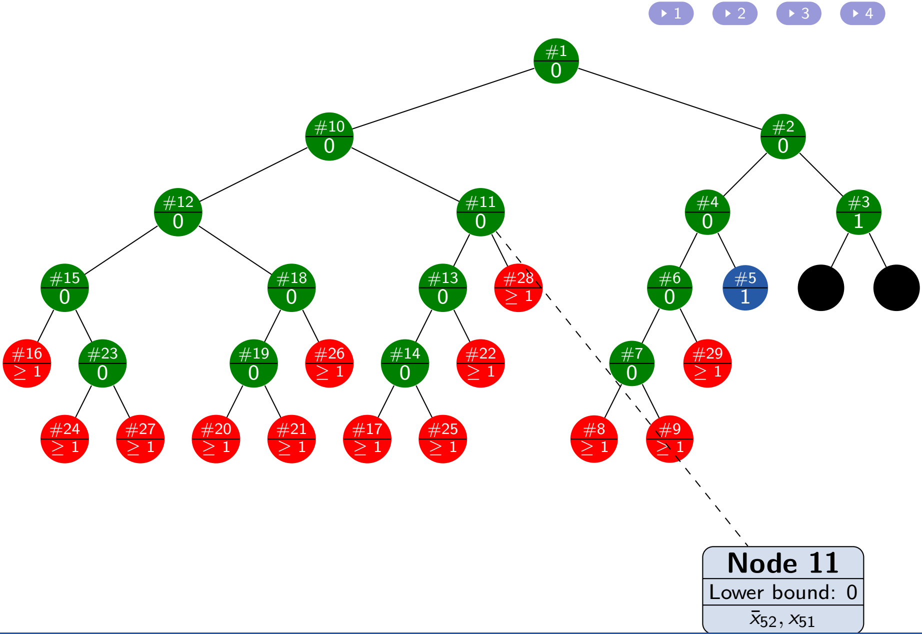
▸ 1 ▸ 2 ▸ 3 ▸ 4


#1
0
#10
0
#2
0
#12
0
#11
0
#4
0
#3
1
#15
0
#18
0
#13
0
#28
≥ 1
#6
0
#5
1
#16
≥ 1
#23
0
#19
0
#26
≥ 1
#14
0
#22
≥ 1
#7
0
#29
≥ 1
#24
≥ 1
#27
≥ 1
#20
≥ 1
#21
≥ 1
#17
≥ 1
#25
≥ 1
#8
≥ 1
#9
≥ 1
Node 9
Lower bound: ≥ 1
$x_{52}, \bar{x}_{53}, \bar{x}_{51}, \bar{x}_{55}, x_{54}$

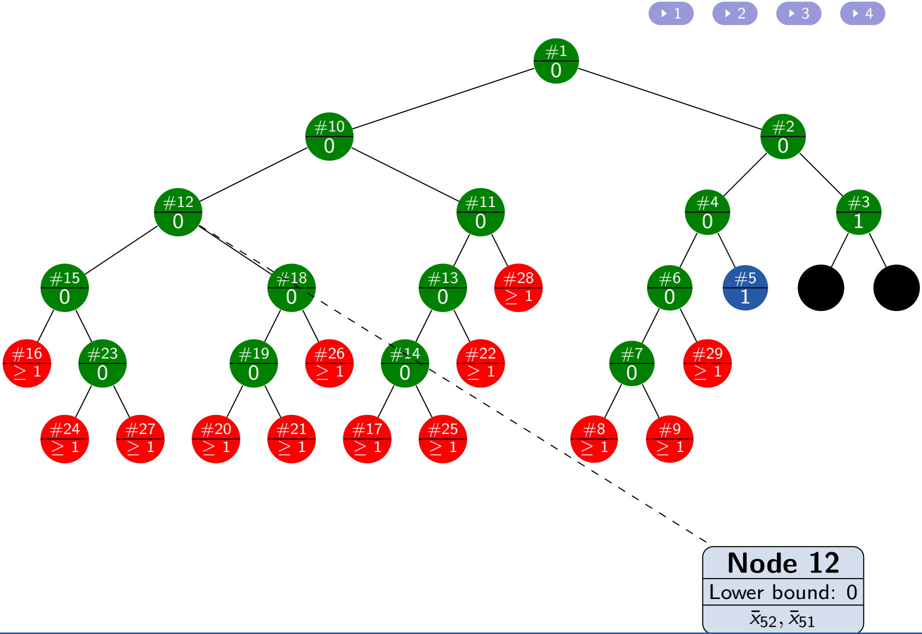
▸ 1 ▸ 2 ▸ 3 ▸ 4



#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 10
Lower bound: 0
$\bar{x}_{52}$


▸ 1
▸ 2
▸ 3
▸ 4
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 11
Lower bound: 0
$\bar{x}_{52}, x_{51}$

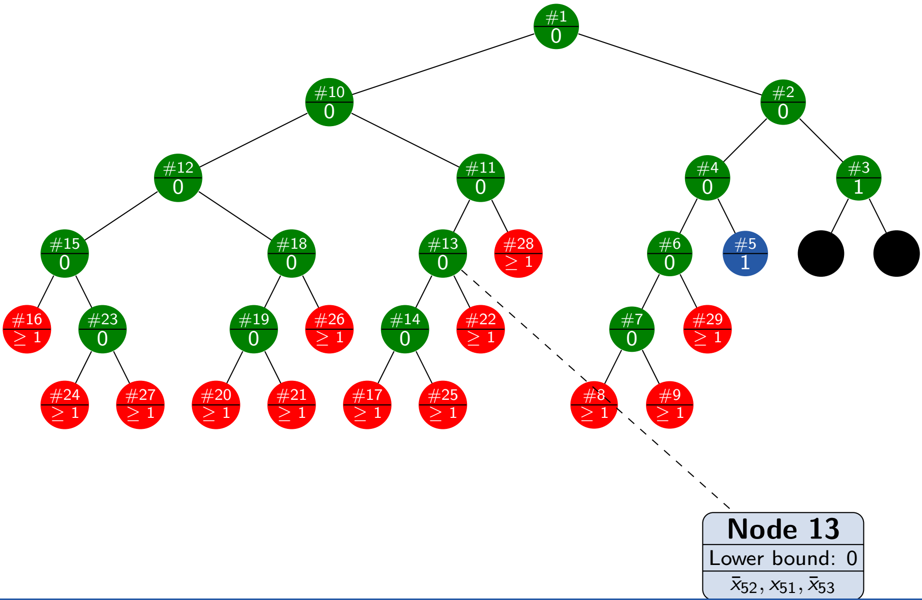
Search tree

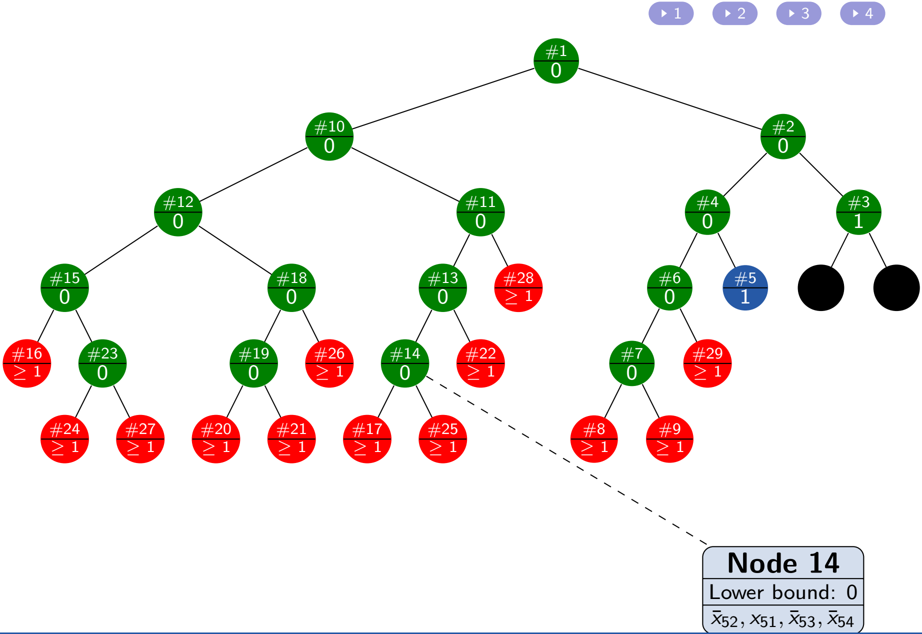


▸ 1
▸ 2
▸ 3
▸ 4
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 12
Lower bound: 0
$\bar{x}_{52}, \bar{x}_{51}$

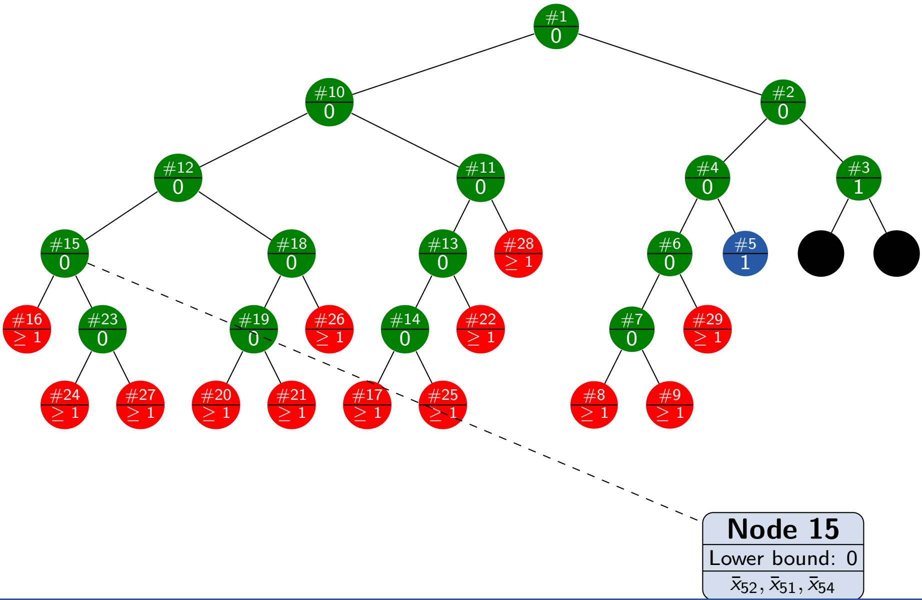
Search tree

1 2 3 4

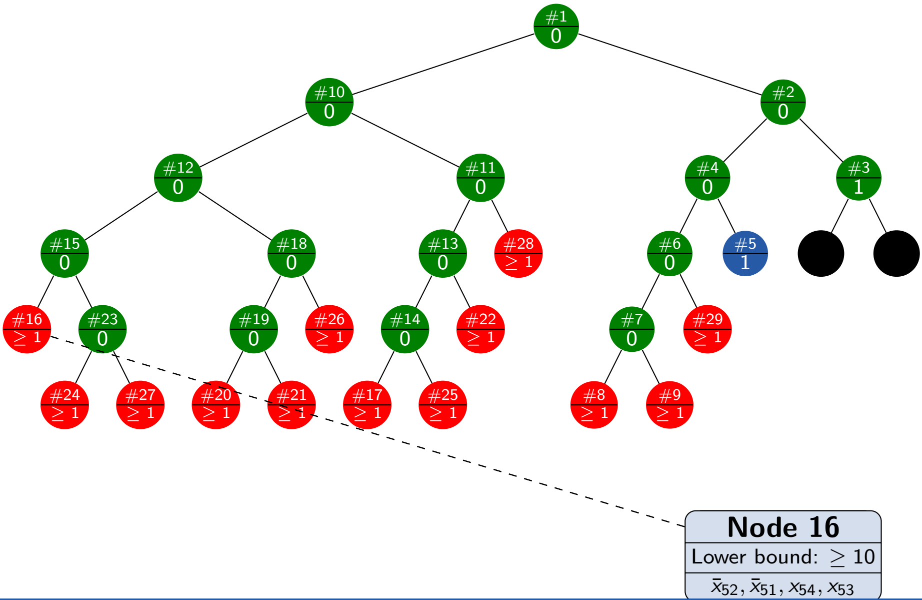



1
2
3
4
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 14
Lower bound: 0
$\bar{x}_{52}, x_{51}, \bar{x}_{53}, \bar{x}_{54}$

▸ 1 ▸ 2 ▸ 3 ▸ 4

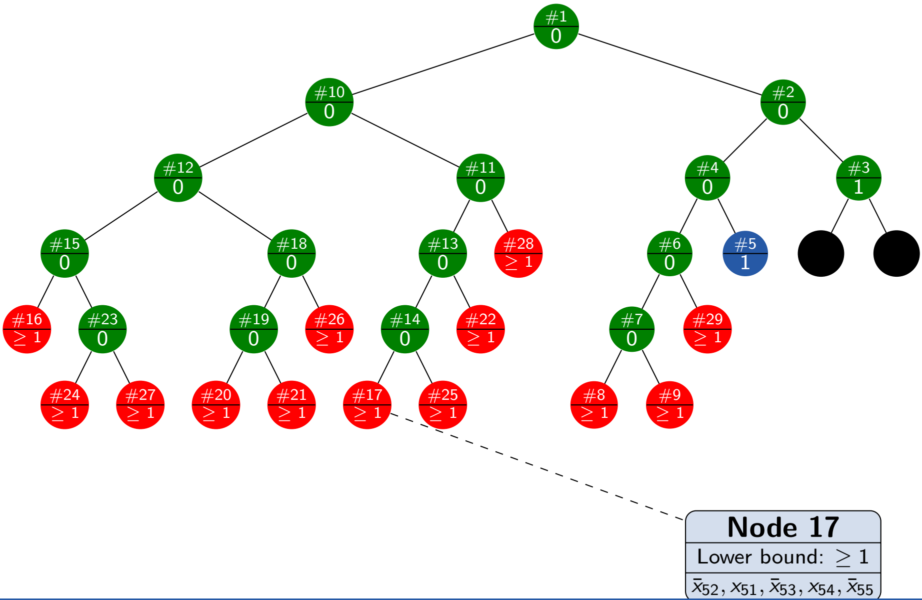

#1
0
#10
0
#2
0
#12
0
#11
0
#4
0
#3
1
#15
0
#18
0
#13
0
#28
≥ 1
#6
0
#5
1
#16
≥ 1
#23
0
#19
0
#26
≥ 1
#14
0
#22
≥ 1
#7
0
#29
≥ 1
#24
≥ 1
#27
≥ 1
#20
≥ 1
#21
≥ 1
#17
≥ 1
#25
≥ 1
#8
≥ 1
#9
≥ 1
Node 15
Lower bound: 0
$\bar{x}_{52}, \bar{x}_{51}, \bar{x}_{54}$

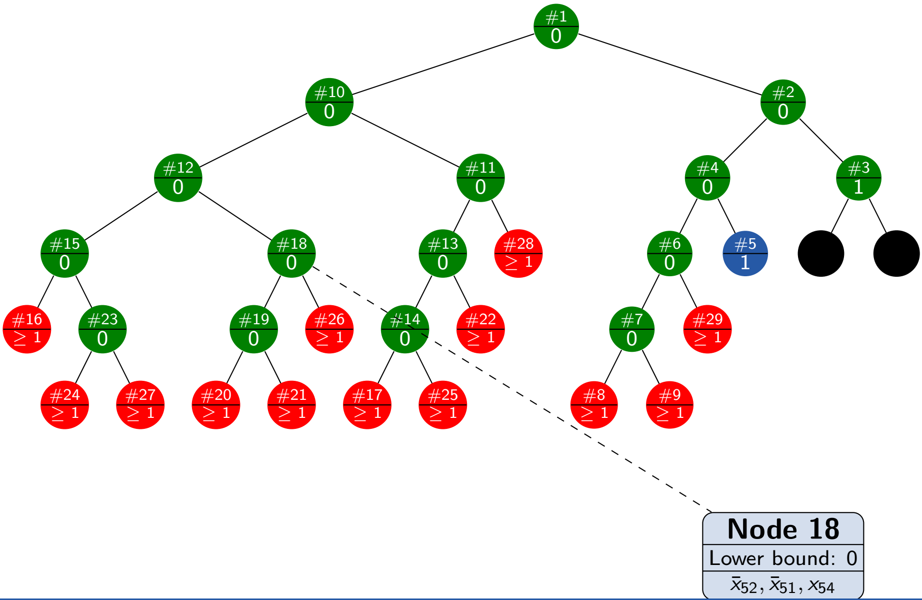
1 2 3 4


#1
0
#10
0
#2
0
#12
0
#11
0
#4
0
#3
1
#15
0
#18
0
#13
0
#28
≥ 1
#6
0
#5
1
#16
≥ 1
#23
0
#19
0
#26
≥ 1
#14
0
#22
≥ 1
#7
0
#29
≥ 1
#24
≥ 1
#27
≥ 1
#20
≥ 1
#21
≥ 1
#17
≥ 1
#25
≥ 1
#8
≥ 1
#9
≥ 1
Node 16
Lower bound: ≥ 10
$\bar{x}_{52}, \bar{x}_{51}, x_{54}, x_{53}$

Search tree

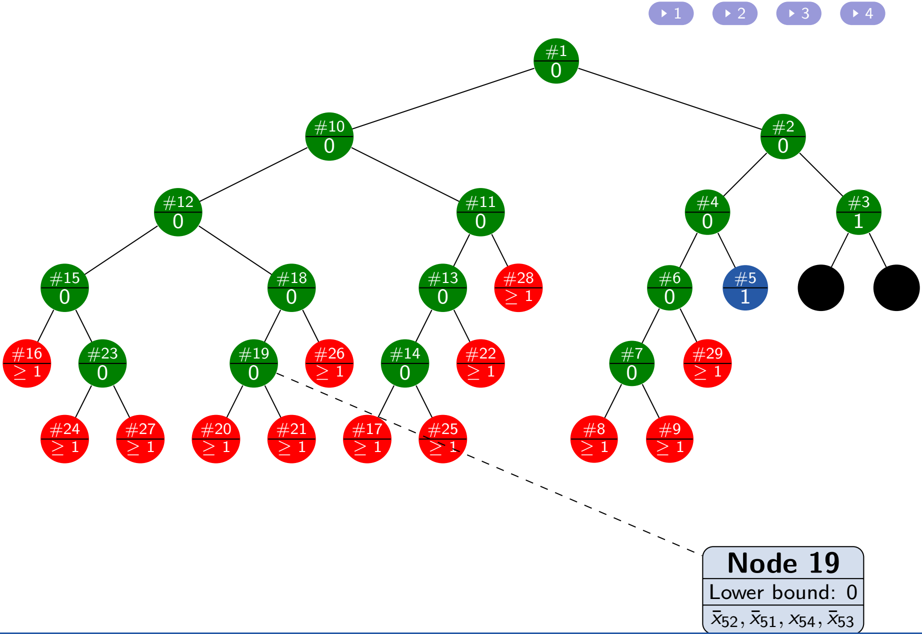
▸ 1 ▸ 2 ▸ 3 ▸ 4



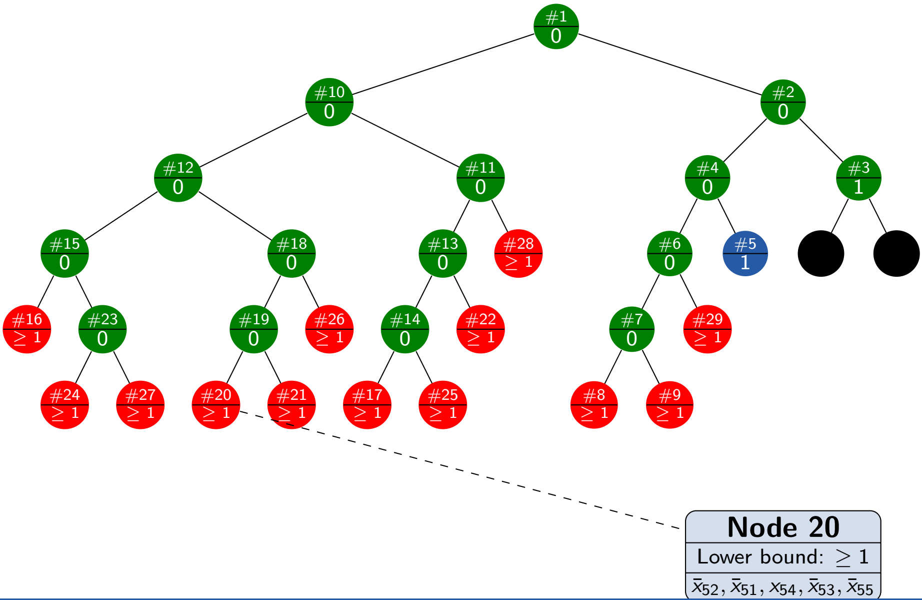


#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 18
Lower bound: 0
$\bar{x}_{52}, \bar{x}_{51}, x_{54}$

Search tree



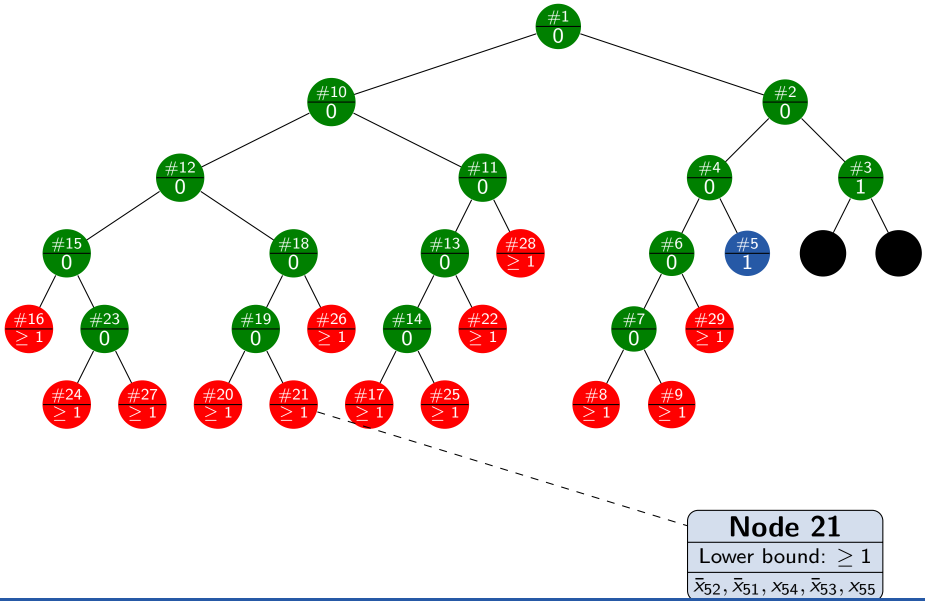
▸ 1 ▸ 2 ▸ 3 ▸ 4
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 19
Lower bound: 0
$\bar{x}_{52}, \bar{x}_{51}, x_{54}, \bar{x}_{53}$



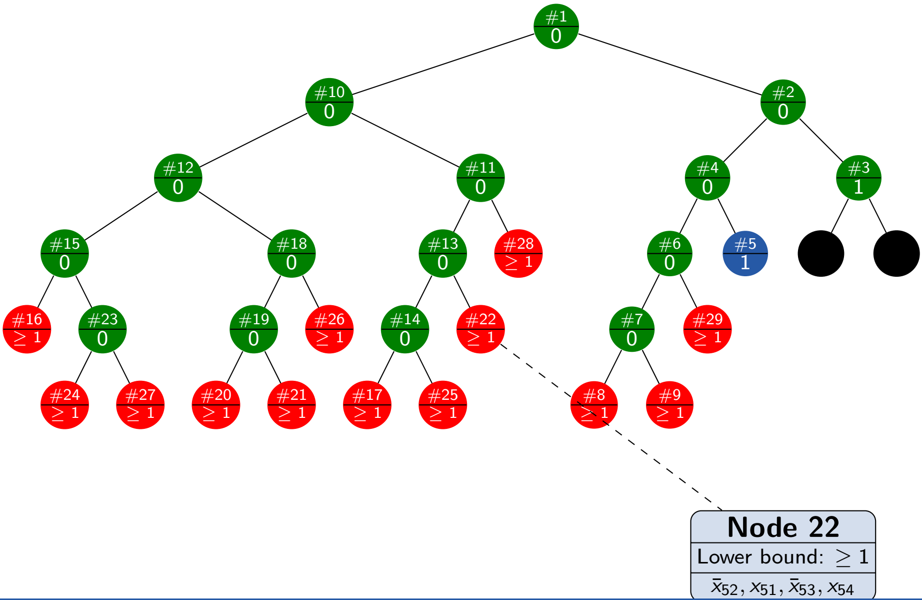
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 20
Lower bound: ≥ 1
$\bar{x}_{52}, \bar{x}_{51}, x_{54}, \bar{x}_{53}, \bar{x}_{55}$

Search tree

▸ 1 ▸ 2 ▸ 3 ▸ 4

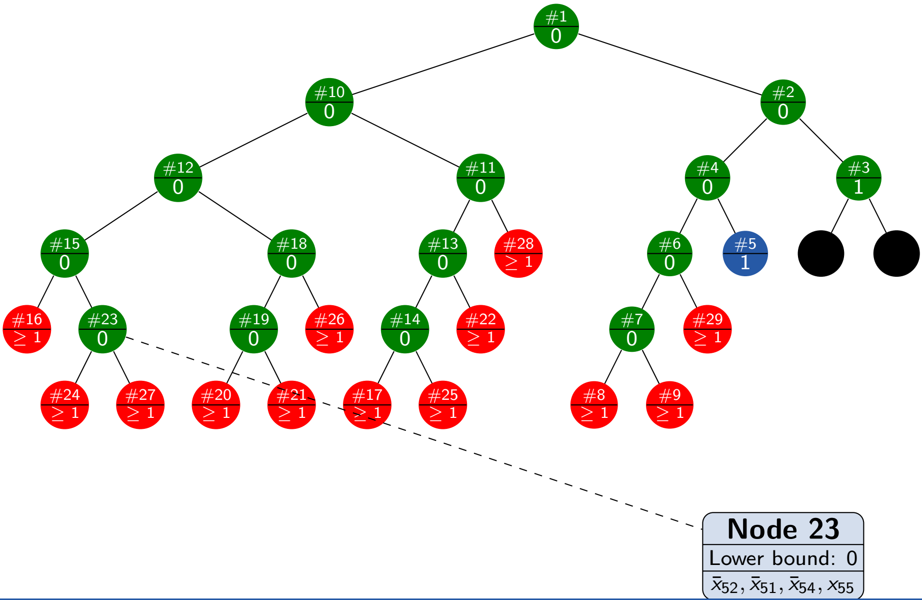


▸ 1 ▸ 2 ▸ 3 ▸ 4

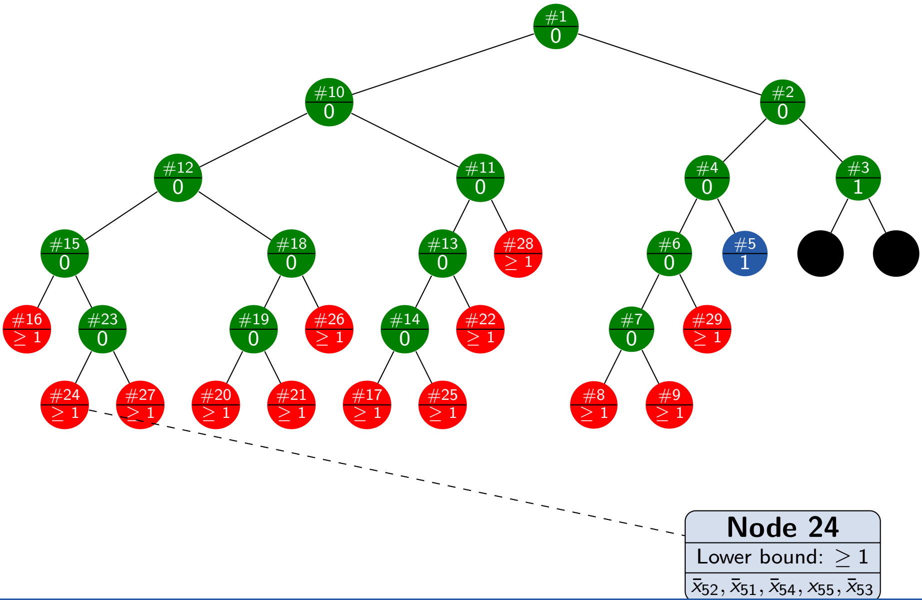


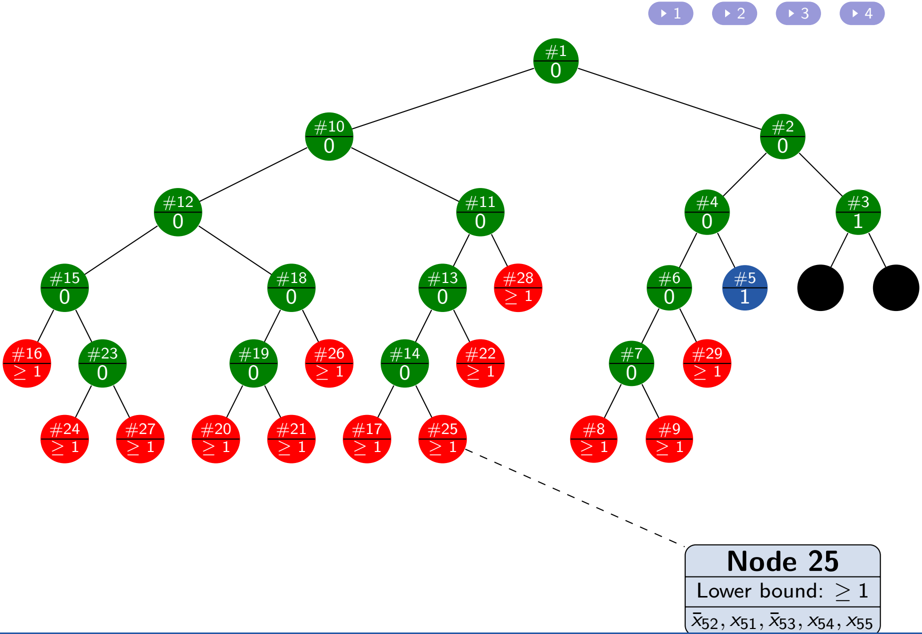
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 22
Lower bound: ≥ 1
$\bar{x}_{52}, x_{51}, \bar{x}_{53}, x_{54}$

▸ 1 ▸ 2 ▸ 3 ▸ 4

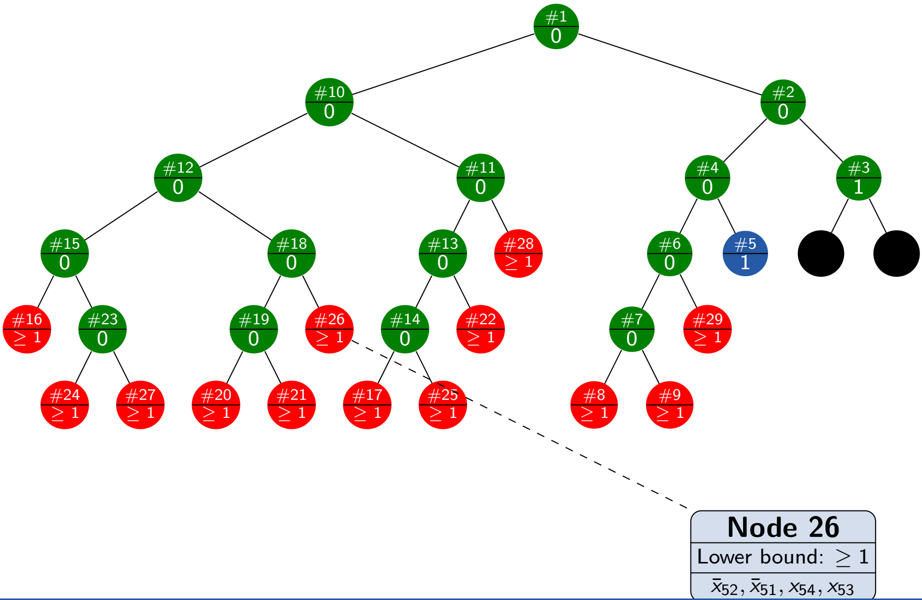

#1
0
#10
0
#2
0
#12
0
#11
0
#4
0
#3
1
#15
0
#18
0
#13
0
#28
≥ 1
#6
0
#5
1
#16
≥ 1
#23
0
#19
0
#26
≥ 1
#14
0
#22
≥ 1
#7
0
#29
≥ 1
#24
≥ 1
#27
≥ 1
#20
≥ 1
#21
≥ 1
#17
≥ 1
#25
≥ 1
#8
≥ 1
#9
≥ 1
Node 23
Lower bound: 0
$\bar{x}_{52}, \bar{x}_{51}, \bar{x}_{54}, x_{55}$

▸ 1 ▸ 2 ▸ 3 ▸ 4


#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 24
Lower bound: ≥ 1
$\bar{x}_{52}, \bar{x}_{51}, \bar{x}_{54}, x_{55}, \bar{x}_{53}$

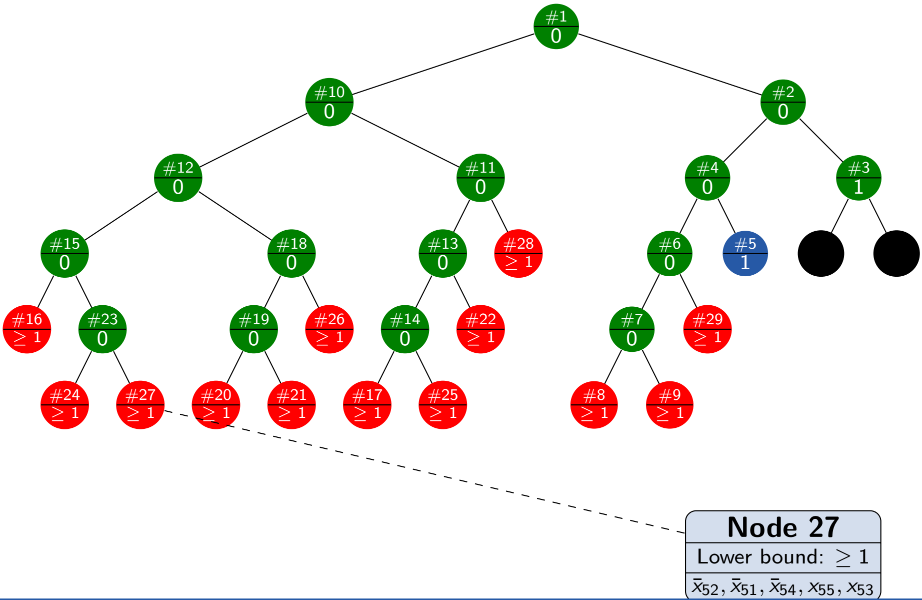

1
2
3
4
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 25
Lower bound: ≥ 1
$\bar{x}_{52}, x_{51}, \bar{x}_{53}, x_{54}, x_{55}$

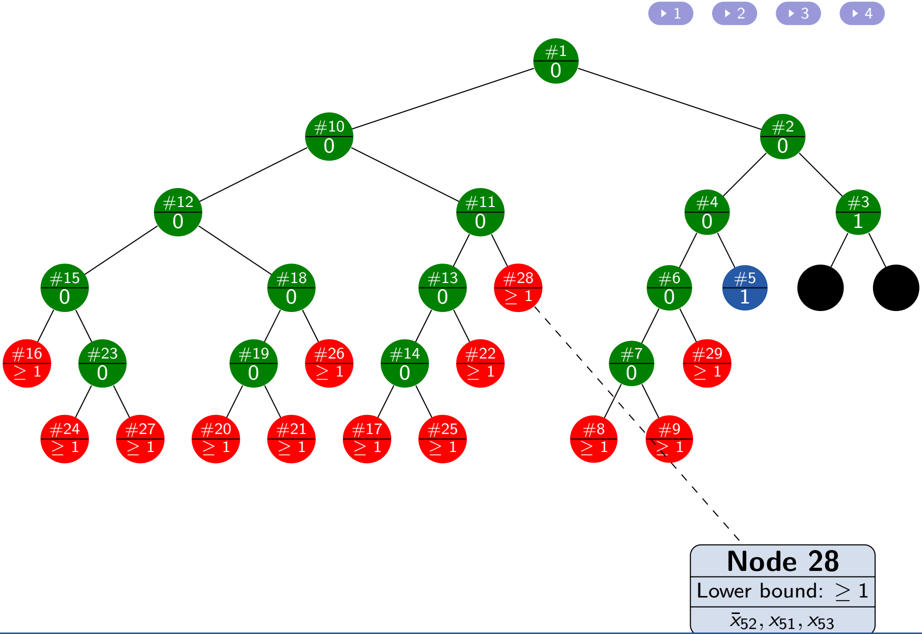
▸ 1 ▸ 2 ▸ 3 ▸ 4


#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 26
Lower bound: ≥ 1
$\bar{x}_{52}, \bar{x}_{51}, x_{54}, x_{53}$

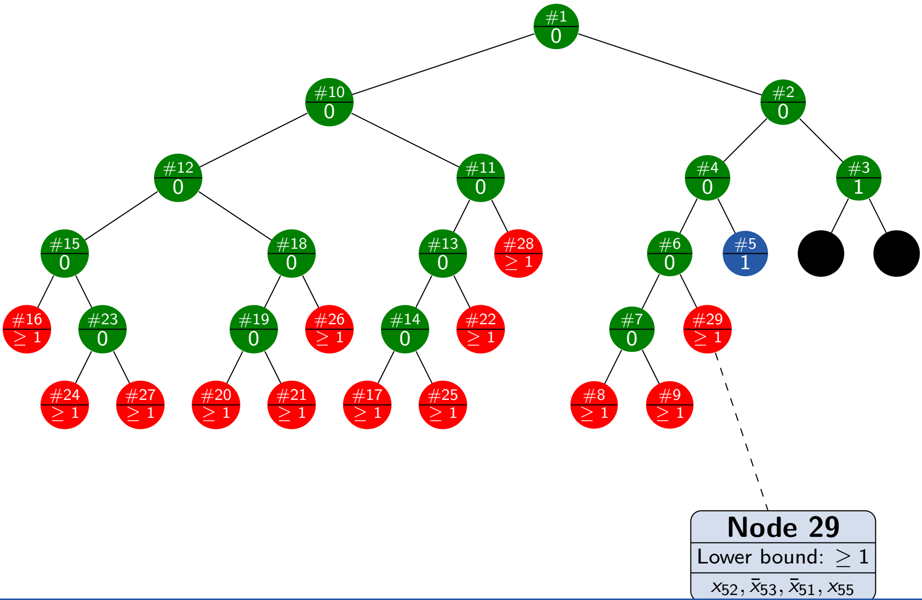
Search tree

▸ 1 ▸ 2 ▸ 3 ▸ 4




1
2
3
4
#1 0
#10 0
#2 0
#12 0
#11 0
#4 0
#3 1
#15 0
#18 0
#13 0
#28 ≥ 1
#6 0
#5 1
#16 ≥ 1
#23 0
#19 0
#26 ≥ 1
#14 0
#22 ≥ 1
#7 0
#29 ≥ 1
#24 ≥ 1
#27 ≥ 1
#20 ≥ 1
#21 ≥ 1
#17 ≥ 1
#25 ≥ 1
#8 ≥ 1
#9 ≥ 1
Node 28
Lower bound: ≥ 1
$\bar{x}_{52}, x_{51}, x_{53}$

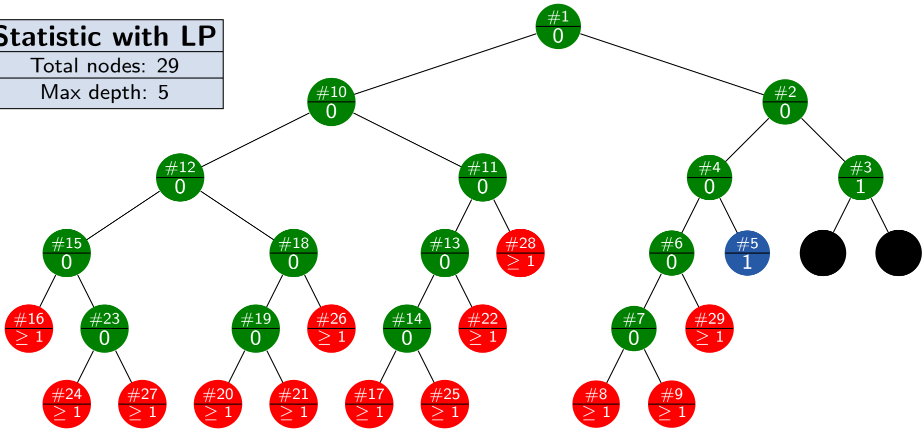
▸ 1 ▸ 2 ▸ 3 ▸ 4


#1
0
#10
0
#2
0
#12
0
#11
0
#4
0
#3
1
#15
0
#18
0
#13
0
#28
≥ 1
#6
0
#5
1
#16
≥ 1
#23
0
#19
0
#26
≥ 1
#14
0
#22
≥ 1
#7
0
#29
≥ 1
#24
≥ 1
#27
≥ 1
#20
≥ 1
#21
≥ 1
#17
≥ 1
#25
≥ 1
#8
≥ 1
#9
≥ 1
Node 29
Lower bound: ≥ 1
$x_{52}, \bar{x}_{53}, \bar{x}_{51}, x_{55}$

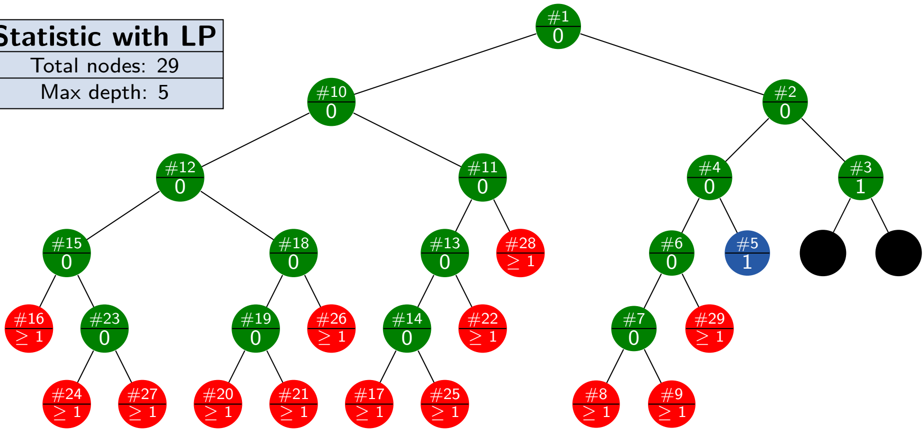
	node	left	depth	frac	curdualbound	dualbound	primalbound
	1	0	0	6	0.000000e+00	0.000000e+00	--
	1	2	0	6	0.000000e+00	0.000000e+00	--
	2	3	1	4	0.000000e+00	0.000000e+00	--
	3	4	2	6	1.000000e+00	0.000000e+00	--
	4	5	2	2	0.000000e+00	0.000000e+00	--
*	5	2	3	-	1.000000e+00	0.000000e+00	1.000000e+00
	6	3	3	2	0.000000e+00	0.000000e+00	1.000000e+00
	7	4	4	2	0.000000e+00	0.000000e+00	1.000000e+00
	8	3	5	-	--	0.000000e+00	1.000000e+00
	9	2	5	-	--	0.000000e+00	1.000000e+00
	10	3	1	2	4.440892e-16	0.000000e+00	1.000000e+00
	11	4	2	2	4.440892e-16	0.000000e+00	1.000000e+00
	12	5	2	4	4.440892e-16	0.000000e+00	1.000000e+00
	13	6	3	4	1.776357e-15	0.000000e+00	1.000000e+00
	14	7	4	4	1.776357e-15	0.000000e+00	1.000000e+00
	node	left	depth	frac	curdualbound	dualbound	primalbound
	15	8	3	4	1.184238e-15	0.000000e+00	1.000000e+00
	16	7	4	-	--	0.000000e+00	1.000000e+00
	17	6	5	-	--	0.000000e+00	1.000000e+00
	18	7	3	2	4.440892e-16	0.000000e+00	1.000000e+00
	19	8	4	4	8.881784e-16	0.000000e+00	1.000000e+00
	20	7	5	-	--	0.000000e+00	1.000000e+00
	21	6	5	-	--	0.000000e+00	1.000000e+00
	22	5	4	-	--	0.000000e+00	1.000000e+00
	23	6	4	2	1.184238e-15	0.000000e+00	1.000000e+00
	24	5	5	-	--	0.000000e+00	1.000000e+00
	25	4	5	-	--	0.000000e+00	1.000000e+00
	26	3	4	-	--	0.000000e+00	1.000000e+00
	27	2	5	-	--	0.000000e+00	1.000000e+00
	28	1	3	-	--	0.000000e+00	1.000000e+00

▸ 1 ▸ 2 ▸ 3 ▸ 4

Statistic with LP
Total nodes: 29
Max depth: 5



Statistic with LP
Total nodes: 29
Max depth: 5



Statistic without LP
Total nodes: 2154
Max depth: 21

Summary

- A solution of a linear relaxation gives a proven dual bound
  - in case of minimization it is a lower bound
  - in case of maximization it is a upper bound
- Linear relaxation is a natural relaxation for an integer program
  - omitting integrality conditions
- Linear relaxation gives a global view w.r.t. all linear constraints
- Linear relaxation guides the search via fractional variables

- A solution of a linear relaxation gives a proven dual bound
  - in case of minimization it is a lower bound
  - in case of maximization it is a upper bound
- Linear relaxation is a natural relaxation for an integer program
  - omitting integrality conditions
- Linear relaxation gives a global view w.r.t. all linear constraints
- Linear relaxation guides the search via fractional variables

Coming up:

- How can a linear program be solved?
- How can an integer program be solved?
- For what is the linear programming relaxation used within an integer programming solver?

Integer Programming for Constraint Programmers

General linear programs (LPs)

Continuous variables:	$x_i \geq 0,\ lb_i \leq x_i \leq ub_i,\ x_i$ free	(columns)
Linear constraints:	$a_1x_1 + \ldots + a_nx_n \overset{\leq}{\underset{\geq}{=}} b$	(rows)
Linear objective:	$c_1x_1 + \ldots + c_nx_n$	$(\rightarrow$ min/max)

General linear programs (LPs)

Continuous variables: $x_i \geq 0$, $lb_i \leq x_i \leq ub_i$, x_i free (columns)

Linear constraints: $a_1 x_1 + \ldots + a_n x_n \begin{smallmatrix} \leq \\ = \\ \geq \end{smallmatrix} b$ (rows)

Linear objective: $c_1 x_1 + \ldots + c_n x_n$ ($\rightarrow$ min/max)

Computational standard form: $\min \left\{ c'x \mid Ax = b, x \geq 0 \right\}$

General linear programs (LPs)

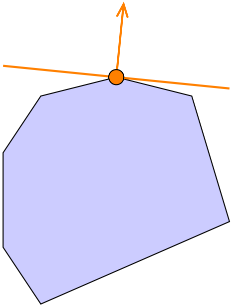
Continuous variables: $x_i \geq 0,\ lb_i \leq x_i \leq ub_i,\ x_i$ free (columns)

Linear constraints: $a_1x_1 + \ldots + a_nx_n \begin{smallmatrix}\leq\\=\\\geq\end{smallmatrix} b$ (rows)

Linear objective: $c_1x_1 + \ldots + c_nx_n$ ($\to$ min/max)

Computational standard form: $\min \left\{ c'x \mid Ax = b, x \geq 0 \right\}$

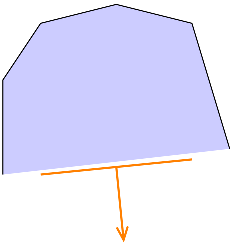
- feasible region is convex and polyhedral
- problem may be
  - unbounded,
  - infeasible, or
  - optimal
- always optimal vertex solution (if an optimal solution exists)



Continuous variables:	$x_i \geq 0,\ lb_i \leq x_i \leq ub_i,\ x_i$ free	(columns)
Linear constraints:	$a_1 x_1 + \ldots + a_n x_n \begin{smallmatrix}\leq\\=\\\geq\end{smallmatrix} b$	(rows)
Linear objective:	$c_1 x_1 + \ldots + c_n x_n$	$(\rightarrow$ min/max)

Computational standard form: $\min \left\{ c'x \mid Ax = b, x \geq 0 \right\}$

- feasible region is convex and polyhedral
- problem may be
  - unbounded,
  - infeasible, or
  - optimal
- always optimal vertex solution (if an optimal solution exists)



General linear programs (LPs)

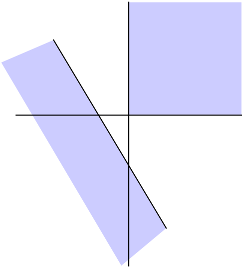
Continuous variables: $x_i \geq 0,\ lb_i \leq x_i \leq ub_i,\ x_i$ free (columns)

Linear constraints: $a_1x_1 + \ldots + a_nx_n \;\substack{\leq\\=\\\geq}\; b$ (rows)

Linear objective: $c_1x_1 + \ldots + c_nx_n$ ($\rightarrow$ min/max)

Computational standard form: $\min \left\{ c'x \mid Ax = b, x \geq 0 \right\}$

- feasible region is convex and polyhedral
- problem may be
  - unbounded,
  - infeasible, or
  - optimal
- always optimal vertex solution (if an optimal solution exists)

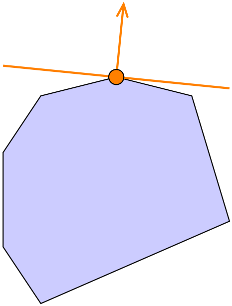


General linear programs (LPs)

Continuous variables:	$x_i \geq 0,\ lb_i \leq x_i \leq ub_i,\ x_i$ free	(columns)
Linear constraints:	$a_1 x_1 + \ldots + a_n x_n \overset{\leq}{\underset{\geq}{=}} b$	(rows)
Linear objective:	$c_1 x_1 + \ldots + c_n x_n$	($\rightarrow$ min/max)

Computational standard form: $\min \left\{ c'x \mid Ax = b, x \geq 0 \right\}$

- feasible region is convex and polyhedral
- problem may be
  - unbounded,
  - infeasible, or
  - optimal
- always optimal vertex solution (if an optimal solution exists)



An incomplete history on linear programming

1827 J. Fourier: Variable elimination algorithm ("Fourier-Motzkin")

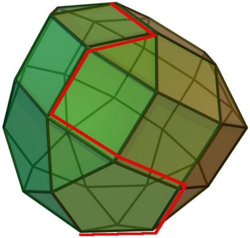
An incomplete history on linear programming

1827 J. Fourier: Variable elimination algorithm ("Fourier-Motzkin")

1947 G. Dantzig: Primal Simplex algorithm

1954 C. Lemke and E. Beale: Dual Simplex algorithm

- by far the most used algorithm to solve LPs
- worst case exponential running time



simplex algorithm
[Dantzig 1947]

An incomplete history on linear programming

1827 J. Fourier: Variable elimination algorithm ("Fourier-Motzkin")

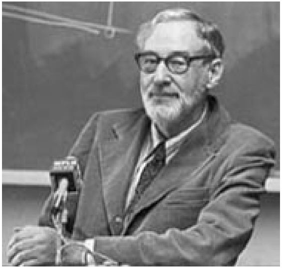
1947 G. Dantzig: Primal Simplex algorithm

1954 C. Lemke and E. Beale: Dual Simplex algorithm

- by far the most used algorithm to solve LPs
- worst case exponential running time

1975 L. Kantorovich and T.C. Koopmans: Nobel prize for Economics

Leonid Kantorovich & Tjalling C. Koopmans



1975: Nobel price in Economic Science

"Optimal allocation of ressources"

An incomplete history on linear programming

1827 J. Fourier: Variable elimination algorithm ("Fourier-Motzkin")

1947 G. Dantzig: Primal Simplex algorithm

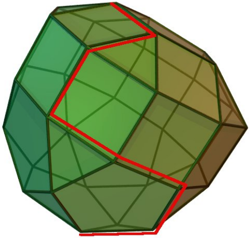
1954 C. Lemke and E. Beale: Dual Simplex algorithm

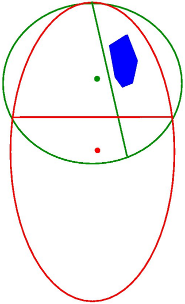
- by far the most used algorithm to solve LPs
- worst case exponential running time

1975 L. Kantorovich and T.C. Koopmans:
Nobel prize for Economics

1979 L. Khachiyan: Ellipsoid Method

- first polynomial time algorithm

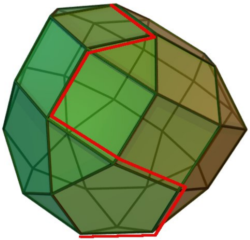




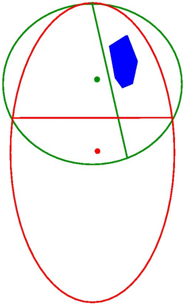
simplex algorithm
[Dantzig 1947]

ellipsoid method
[Khachiyan 1979]

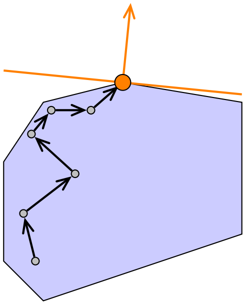
LP algorithms



simplex algorithm
[Dantzig 1947]



ellipsoid method
[Khachiyan 1979]



interior point
[Karmarkar 1984]

An incomplete history on linear programming

1827 J. Fourier: Variable elimination algorithm ("Fourier-Motzkin")

1947 G. Dantzig: Primal Simplex algorithm

1954 C. Lemke and E. Beale: Dual Simplex algorithm

- by far the most used algorithm to solve LPs
- worst case exponential running time

1975 L. Kantorovich and T.C. Koopmans:
Nobel prize for Economics

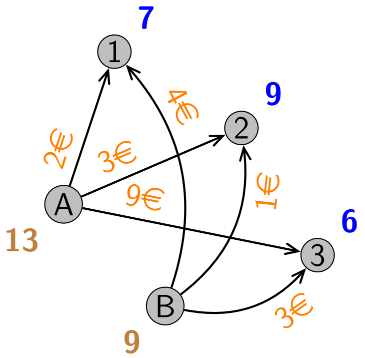
1979 L. Khachiyan: Ellipsoid Method

- first polynomial time algorithm

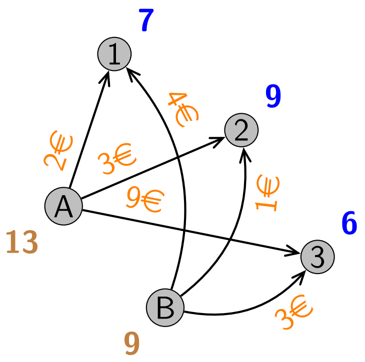
1984 N. Karmarkar: Interior Point Method/Barrier Algorithm

$\geq$ 1987 Primal-Dual Interior Point Algorithms

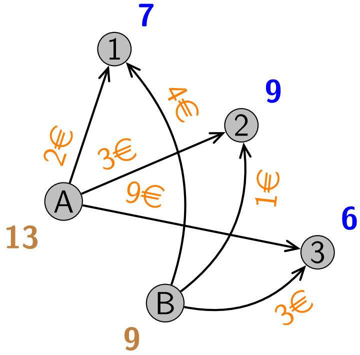
- basis for state-of-the-art interior point implementations
- for single, sparse LPs often faster than simplex



7
1
9
2
6
3
A
13
B
9
2€
3€
9€
4€
1€
3€



$$
\begin{aligned}
\min \quad & 2x_{A,1} + 3x_{A,2} + 9x_{A,3} + \\
& 4x_{B,1} + 1x_{B,2} + 3x_{B,3} \\
\text{s.t.} \quad & x_{A,1} + x_{A,2} + x_{A,3} = 13 \\
& x_{B,1} + x_{B,2} + x_{B,3} = 9 \\
& x_{A,1} + x_{B,1} = 7 \\
& x_{A,2} + x_{B,2} = 9 \\
& x_{A,3} + x_{B,3} = 6 \\
& x \geq 0
\end{aligned}
$$

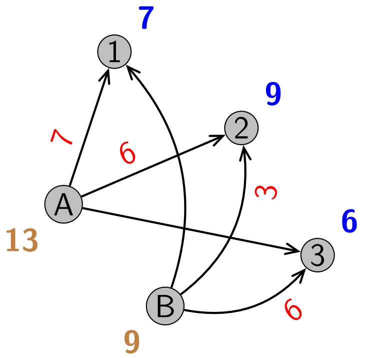


$$\min \quad 2x_{A,1} + 3x_{A,2} + 9x_{A,3} + 4x_{B,1} + 1x_{B,2} + 3x_{B,3}$$

$$\begin{aligned} \text{s.t.} \quad x_{A,1} + x_{A,2} + x_{A,3} &= 13 \\ x_{B,1} + x_{B,2} + x_{B,3} &= 9 \\ x_{A,1} + x_{B,1} &= 7 \\ x_{A,2} + x_{B,2} &= 9 \\ x_{A,3} + x_{B,3} &= 6 \\ x &\geq 0 \end{aligned}$$

Heuristic solution with objective value 53:

$A \to 1 = 7 \qquad A \to 2 = 6 \qquad B \to 2 = 3 \qquad B \to 3 = 6$



$$\begin{aligned}
\min \quad & 2x_{A,1} + 3x_{A,2} + 9x_{A,3} + \\
& 4x_{B,1} + 1x_{B,2} + 3x_{B,3} \\
\text{s.t.} \quad & x_{A,1} + x_{A,2} + x_{A,3} = 13 \\
& x_{B,1} + x_{B,2} + x_{B,3} = 9 \\
& x_{A,1} + x_{B,1} = 7 \\
& x_{A,2} + x_{B,2} = 9 \\
& x_{A,3} + x_{B,3} = 6 \\
& x \geq 0
\end{aligned}$$

Heuristic solution with objective value 53:

$A \rightarrow 1 = 7$ $\quad A \rightarrow 2 = 6$ $\quad B \rightarrow 2 = 3$ $\quad B \rightarrow 3 = 6$

Dual multipliers: proofing solution quality

$$
\begin{array}{lrrrrrrl}
\min & 2x_{A,1} & +3x_{A,2} & +9x_{A,3} & +4x_{B,1} & +1x_{B,2} & +3x_{B,3} & \\
\text{s.t.} & x_{A,1} & +x_{A,2} & +x_{A,3} & & & & =13 \\
& & & & x_{B,1} & +x_{B,2} & +x_{B,3} & =9 \\
& x_{A,1} & & & +x_{B,1} & & & =7 \\
& & x_{A,2} & & & +x_{B,2} & & =9 \\
& & & x_{A,3} & & & +x_{B,3} & =6 \\
& & & & & & x & \geq 0
\end{array}
$$

Dual multipliers: proofing solution quality

$$
\begin{array}{llcl}
\min & & 2x_{A,1}+3x_{A,2}+9x_{A,3}+4x_{B,1}+1x_{B,2}+3x_{B,3} & \\
\text{s.t.} & \color{red}{2\times} & x_{A,1}+x_{A,2}+x_{A,3} & =13 \\
& \color{red}{1\times} & x_{B,1}+x_{B,2}+x_{B,3} & =9 \\
& & x_{A,1}+x_{B,1} & =7 \\
& & x_{A,2}+x_{B,2} & =9 \\
& \color{red}{2\times} & x_{A,3}+x_{B,3} & =6 \\
& & x & \geq 0 \\
& \color{red}{+} & & \\
\hline
\color{red}{\Rightarrow \min \geq} & & 2x_{A,1}+2x_{A,2}+3x_{A,3}+1x_{B,1}+1x_{B,2}+3x_{B,3} & =\mathbf{47}
\end{array}
$$

$$
\begin{array}{llcccccccl}
\min & & 2x_{A,1} & +3x_{A,2} & +9x_{A,3} & +4x_{B,1} & +1x_{B,2} & +3x_{B,3} & & \\
\text{s.t.} & \color{red}{3\times} & x_{A,1} & +x_{A,2} & +x_{A,3} & & & & = & 13 \\
 & \color{red}{1\times} & & & & x_{B,1} & +x_{B,2} & +x_{B,3} & = & 9 \\
 & \color{red}{-1\times} & x_{A,1} & & & +x_{B,1} & & & = & 7 \\
 & & & x_{A,2} & & & +x_{B,2} & & = & 9 \\
 & \color{red}{2\times} & & & x_{A,3} & & & +x_{B,3} & = & 6 \\
 & & & & & & & x & \geq & 0 \\
 & \color{red}{+} & & & & & & & & \\
\hline
\color{red}{\Rightarrow \min \geq} & & 2x_{A,1} & +3x_{A,2} & +5x_{A,3} & & +1x_{B,2} & +3x_{B,3} & = & \mathbf{53}
\end{array}
$$

Solution is optimal!

Primal LP

$\min \left\{ c^T x \mid Ax = b, x \geq 0 \right\}$

Dual LP

$\max \left\{ b^T y \mid y^T A \leq c^T, y \in \mathbb{R}^m \right\}$

Simple observation: For any x, y feasible,

$$c^T x \geq y^T A x = b^T y.$$

Primal LP

$\min \{ c^T x \mid Ax = b, x \geq 0 \}$

Dual LP

$\max \{ b^T y \mid y^T A \leq c^T, y \in \mathbb{R}^m \}$

Simple observation: For any x, y feasible,

$$c^T x \geq y^T A x = b^T y.$$

- **Weak Duality:**

$$\min \{ c^T x \mid Ax = b, x \geq 0 \} \geq \max \{ b^T y \mid y^T A \leq c^T, y \in \mathbb{R}^m \}$$

Primal LP

$$\min \left\{ c^T x \mid Ax = b, x \geq 0 \right\}$$

Dual LP

$$\max \left\{ b^T y \mid y^T A \leq c^T, y \in \mathbb{R}^m \right\}$$

Simple observation: For any x, y feasible,

$$c^T x \geq y^T A x = b^T y.$$

- **Weak Duality:**

$$\min \left\{ c^T x \mid Ax = b, x \geq 0 \right\} \geq \max \left\{ b^T y \mid y^T A \leq c^T, y \in \mathbb{R}^m \right\}$$

- **Strong Duality:**

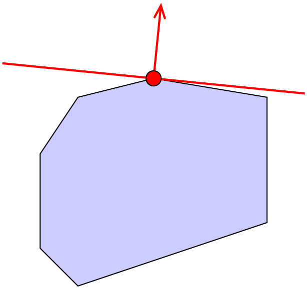
$$\min \left\{ c^T x \mid Ax = b, x \geq 0 \right\} = \max \left\{ b^T y \mid y^T A \leq c^T, y \in \mathbb{R}^m \right\}$$

Consider n variables, m constraints:

$$\min \left\{ c^T x \mid Ax = b, x \geq 0 \right\}$$

with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

- If optimal: there always exists an optimal vertex solution.

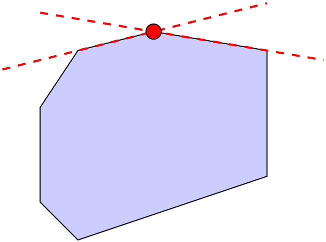


Consider n variables, m constraints:

$$\min \{ c^T x \mid Ax = b, x \geq 0 \}$$

with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

- If optimal: there always exists an optimal vertex solution.
- Vertices uniquely determined by n tight inequalities:



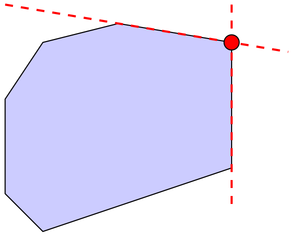
- In standard from that means
  - m equality constraints $Ax = b$
  - $n - m$ tight bounds $x_i = 0$

Consider n variables, m constraints:

$$\min \left\{ c^T x \mid Ax = b, x \geq 0 \right\}$$

with $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$.

- If optimal: there always exists an optimal vertex solution.
- Vertices uniquely determined by n tight inequalities:



- In standard from that means
  - m equality constraints $Ax = b$
  - $n - m$ tight bounds $x_i = 0$

Primal solution

- Fix $n - m$ variables: $x_i = 0$ for $i \in \mathcal{N} \subseteq \{1, \ldots, n\}$
- m variables remain: x_i for $i \in \mathcal{B} = \{1, \ldots, n\} \setminus \mathcal{N}$
- Solve linear system with m equations, m variables:

$$Ax = b \ \rightsquigarrow \ A_{\mathcal{B}} x_{\mathcal{B}} = b \ \rightsquigarrow \ x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$$

Primal solution

- Fix $n - m$ variables: $x_i = 0$ for $i \in \mathcal{N} \subseteq \{1, \ldots, n\}$
- m variables remain: x_i for $i \in \mathcal{B} = \{1, \ldots, n\} \setminus \mathcal{N}$
- Solve linear system with m equations, m variables:

$$Ax = b \rightsquigarrow A_{\mathcal{B}} x_{\mathcal{B}} = b \rightsquigarrow x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$$

Dual multipliers

- Globally: find y such that $y^T A \leq c^T$
- Locally: ignore fixed variables and solve

$$y^T A_{\mathcal{B}} = c_{\mathcal{B}}^T \rightsquigarrow y^T = c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}$$

Primal solution

- Fix $n - m$ variables: $x_i = 0$ for $i \in \mathcal{N} \subseteq \{1, \ldots, n\}$
- m variables remain: x_i for $i \in \mathcal{B} = \{1, \ldots, n\} \setminus \mathcal{N}$
- Solve linear system with m equations, m variables:

$$Ax = b \rightsquigarrow A_{\mathcal{B}} x_{\mathcal{B}} = b \rightsquigarrow x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$$

Dual multipliers

- Globally: find y such that $y^T A \leq c^T$
- Locally: ignore fixed variables and solve

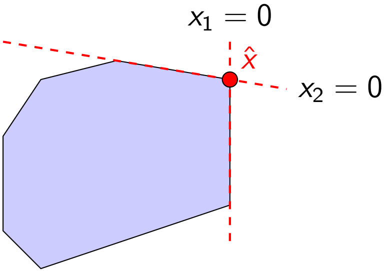
$$y^T A_{\mathcal{B}} = c_{\mathcal{B}}^T \rightsquigarrow y^T = c_{\mathcal{B}}^T A_{\mathcal{B}}^{-1}$$

Basic solution = discrete basis $\mathcal{B}$ + primal sol. x + dual mult. y

- In theory: could enumerate $\binom{n}{m}$ basic solutions.

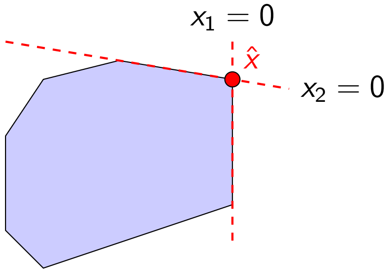
- $x_\mathcal{B}$ is a function of $x_\mathcal{N}$:

$$A_\mathcal{B} x_\mathcal{B} + A_\mathcal{N} x_\mathcal{N} = b$$



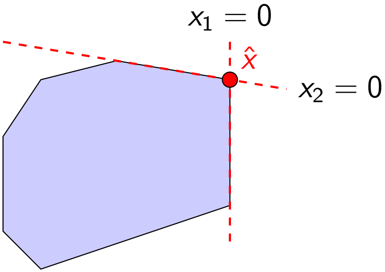
- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

$$A_{\mathcal{B}} x_{\mathcal{B}} = b - A_{\mathcal{N}} x_{\mathcal{N}}$$



- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

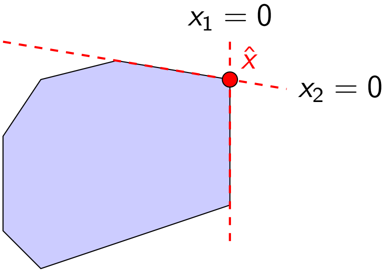
$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}} x_{\mathcal{N}})$$



- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}} x_{\mathcal{N}})$$

- $\hat{x}_{\mathcal{N}} = 0 \;\rightsquigarrow\; \hat{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$

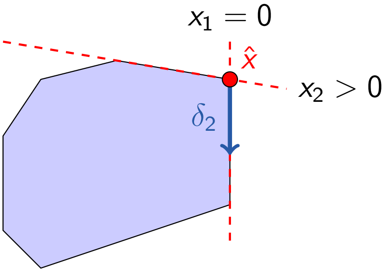


- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}} x_{\mathcal{N}})$$

- $\hat{x}_{\mathcal{N}} = 0 \;\rightsquigarrow\; \hat{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$
- Unfix x_i, $i \in \mathcal{N}$:

$$x_{\mathcal{B}} = \hat{x}_{\mathcal{B}} - \underbrace{A_{\mathcal{B}}^{-1} A_i}_{\delta_i} x_i$$

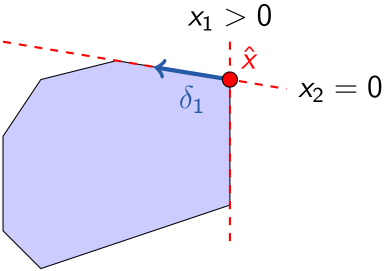


- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}} x_{\mathcal{N}})$$

- $\hat{x}_{\mathcal{N}} = 0 \ \rightsquigarrow \ \hat{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$
- Unfix x_i, $i \in \mathcal{N}$:

$$x_{\mathcal{B}} = \hat{x}_{\mathcal{B}} - \underbrace{A_{\mathcal{B}}^{-1} A_i}_{\delta_i} x_i$$



- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

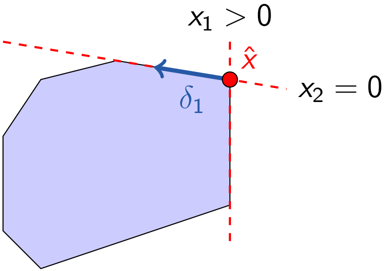
$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}} x_{\mathcal{N}})$$

- $\hat{x}_{\mathcal{N}} = 0 \;\rightsquigarrow\; \hat{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$
- Unfix x_i, $i \in \mathcal{N}$:

$$x_{\mathcal{B}} = \hat{x}_{\mathcal{B}} - \underbrace{A_{\mathcal{B}}^{-1} A_i}_{\delta_i} x_i$$

- Objective change:

$$c^T x = c^T \hat{x} + c_i x_i - c^T \delta_i x_i$$

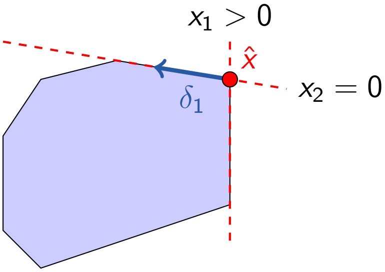


- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}} x_{\mathcal{N}})$$

- $\hat{x}_{\mathcal{N}} = 0 \rightsquigarrow \hat{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$
- Unfix x_i, $i \in \mathcal{N}$:

$$x_{\mathcal{B}} = \hat{x}_{\mathcal{B}} - \underbrace{A_{\mathcal{B}}^{-1} A_i}_{\delta_i} x_i$$



- Objective change:

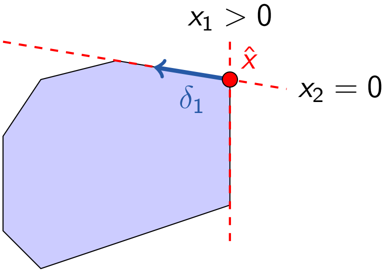
$$c^T x = c^T \hat{x} + c_i x_i - c^T \delta_i x_i = c^T \hat{x} + \underbrace{(c_i - y^T A_i)}_{\text{reduced cost } r_i} x_i$$

- $x_{\mathcal{B}}$ is a function of $x_{\mathcal{N}}$:

$$x_{\mathcal{B}} = A_{\mathcal{B}}^{-1}(b - A_{\mathcal{N}} x_{\mathcal{N}})$$

- $\hat{x}_{\mathcal{N}} = 0 \rightsquigarrow \hat{x}_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b$
- Unfix x_i, $i \in \mathcal{N}$:

$$x_{\mathcal{B}} = \hat{x}_{\mathcal{B}} - \underbrace{A_{\mathcal{B}}^{-1} A_i}_{\delta_i} x_i$$



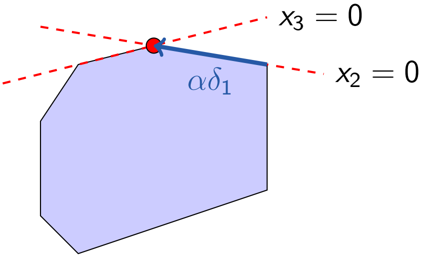
- Objective change:

$$c^T x = c^T \hat{x} + c_i x_i - c^T \delta_i x_i = c^T \hat{x} + \underbrace{(c_i - y^T A_i)}_{\text{reduced cost } r_i} x_i$$

- **Reduced cost** of x_i = **obj. change per unit increase** of x_i

- $x_\mathcal{B}$ is a function of $x_\mathcal{N}$:

$$x_\mathcal{B} = A_\mathcal{B}^{-1}(b - A_\mathcal{N} x_\mathcal{N})$$



- $\hat{x}_\mathcal{N} = 0 \rightsquigarrow \hat{x}_\mathcal{B} = A_\mathcal{B}^{-1} b$
- Unfix x_i, $i \in \mathcal{N}$:

$$x_\mathcal{B} = \hat{x}_\mathcal{B} - \underbrace{A_\mathcal{B}^{-1} A_i}_{\delta_i} x_i$$

- Objective change:

$$c^T x = c^T \hat{x} + c_i x_i - c^T \delta_i x_i = c^T \hat{x} + \underbrace{(c_i - y^T A_i)}_{\text{reduced cost } r_i} x_i$$

- **Reduced cost** of x_i = **obj. change per unit increase** of x_i

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b \geq 0$,

Primal simplex algorithm

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b \geq 0$,

- ▷ maintain primal feasibility
- ▷ improve obj. value until reduced costs are $\geq 0 \Leftrightarrow$ dual feasible

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b \geq 0$,

- ▷ maintain primal feasibility
- ▷ improve obj. value until reduced costs are $\geq 0 \Leftrightarrow$ dual feasible

Init

- ▷ factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"

Primal simplex algorithm

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b \geq 0$,

- maintain primal feasibility
- improve obj. value until reduced costs are $\geq 0 \Leftrightarrow$ dual feasible

Init

- factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"
- solve for $x_{\mathcal{B}}$ and y
- compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_\mathcal{B} = A_\mathcal{B}^{-1} b \geq 0$,

- ▷ maintain primal feasibility
- ▷ improve obj. value until reduced costs are $\geq 0 \Leftrightarrow$ dual feasible

Init

- ▷ factorize basis matrix $A_\mathcal{B} \rightsquigarrow$ "$A_\mathcal{B}^{-1}$"
- ▷ solve for $x_\mathcal{B}$ and y
- ▷ compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Repeat

- ▷ if $r_i \geq 0$ for all $i \in \mathcal{N}$ stop $\rightsquigarrow$ OPTIMAL

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b \geq 0$,

- ▷ maintain primal feasibility
- ▷ improve obj. value until reduced costs are $\geq 0 \Leftrightarrow$ dual feasible

Init

- ▷ factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"
- ▷ solve for $x_{\mathcal{B}}$ and y
- ▷ compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Repeat

- ▷ if $r_i \geq 0$ for all $i \in \mathcal{N}$ stop $\rightsquigarrow$ OPTIMAL
  else choose $r_i < 0$, $i \in \mathcal{N}$ (pricing)

Primal simplex algorithm

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b \geq 0$,

- ▷ maintain primal feasibility
- ▷ improve obj. value until reduced costs are $\geq 0 \Leftrightarrow$ dual feasible

Init

- ▷ factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"
- ▷ solve for $x_{\mathcal{B}}$ and y
- ▷ compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Repeat

- ▷ if $r_i \geq 0$ for all $i \in \mathcal{N}$ stop $\rightsquigarrow$ OPTIMAL
  else choose $r_i < 0$, $i \in \mathcal{N}$ (pricing)
- ▷ compute max. steplength α s.t. $x_{\mathcal{B}} - \alpha \delta_i \geq 0$ (ratiotest)

Idea: Given a primal feasible starting basis $\mathcal{B}$, i.e., $x_{\mathcal{B}} = A_{\mathcal{B}}^{-1} b \geq 0$,

- ▷ maintain primal feasibility
- ▷ improve obj. value until reduced costs are $\geq 0 \Leftrightarrow$ dual feasible

Init

- ▷ factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"
- ▷ solve for $x_{\mathcal{B}}$ and y
- ▷ compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Repeat

- ▷ if $r_i \geq 0$ for all $i \in \mathcal{N}$ stop $\rightsquigarrow$ OPTIMAL
  else choose $r_i < 0$, $i \in \mathcal{N}$ (pricing)
- ▷ compute max. steplength α s.t. $x_{\mathcal{B}} - \alpha\delta_i \geq 0$ (ratiotest)
- ▷ if $\alpha = \infty$ stop $\rightsquigarrow$ UNBOUNDED
  else update $\mathcal{B}$, x, y, r, $A_{\mathcal{B}}^{-1}$

Re-solving and hot starts

An IP solver classically solves many related LPs.

▷ **modified objective function**

▷ **changed variable bounds** or **added constraints**

Re-solving and hot starts

An IP solver classically solves many related LPs.

- **modified objective function**
  - last basic solution becomes suboptimal, but remains primal feasible
  - only reduced costs $r_i = c_i - y^T A_i$ change
  - continue with primal simplex iterations
- **changed variable bounds** or **added constraints**

Re-solving and hot starts

An IP solver classically solves many related LPs.

- **modified objective function**
  - last basic solution becomes suboptimal, but remains primal feasible
  - only reduced costs $r_i = c_i - y^T A_i$ change
  - continue with primal simplex iterations
- **changed variable bounds** or **added constraints**
  - last basic solution becomes primal infeasible
  - dual multipliers and reduced costs unchanged
  - continue with dual simplex

Re-solving and hot starts

An IP solver classically solves many related LPs.

- **modified objective function**
  - last basic solution becomes suboptimal, but remains primal feasible
  - only reduced costs $r_i = c_i - y^T A_i$ change
  - continue with primal simplex iterations
- **changed variable bounds** or **added constraints**
  - last basic solution becomes primal infeasible
  - dual multipliers and reduced costs unchanged
  - continue with dual simplex

Dual simplex

- basic procedures as in primal simplex
- maintains dual feasibility and moves towards primal feasibility
- objective value increases towards optimum
- typically **very** few iterations to re-optimize

Dual simplex algorithm

Idea: Given a dual feasible starting basis $\mathcal{B}$, i.e., $r \geq 0$,

- ▷ maintain dual feasibility
- ▷ reduce primal infeasibility until $x \geq 0$

Idea: Given a dual feasible starting basis $\mathcal{B}$, i.e., $r \geq 0$,

- maintain dual feasibility
- reduce primal infeasibility until $x \geq 0$

Init

- factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"
- solve for $x_{\mathcal{B}}$ and y
- compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Idea: Given a dual feasible starting basis $\mathcal{B}$, i.e., $r \geq 0$,

- maintain dual feasibility
- reduce primal infeasibility until $x \geq 0$

Init

- factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"
- solve for $x_{\mathcal{B}}$ and y
- compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Repeat

- if $x_i \geq 0$ for all $i \in \mathcal{B}$ stop $\rightsquigarrow$ OPTIMAL
  else choose $x_i < 0,\ i \in \mathcal{B}$ (pricing)
- compute max. steplength α (ratiotest)
- if $\alpha = \infty$ stop $\rightsquigarrow$ INFEASIBLE
  else update $\mathcal{B}$, x, y, r, $A_{\mathcal{B}}^{-1}$

Idea: Given a dual feasible starting basis $\mathcal{B}$, i.e., $r \geq 0$,

- maintain dual feasibility
- reduce primal infeasibility until $x \geq 0$

Init

- factorize basis matrix $A_{\mathcal{B}} \rightsquigarrow$ "$A_{\mathcal{B}}^{-1}$"
- solve for $x_{\mathcal{B}}$ and y
- compute reduced costs: $r_i = c_i - y^T A_i$ for $i \in \mathcal{N}$

Repeat while $c^T x < z^*$ obj. limit

- if $x_i \geq 0$ for all $i \in \mathcal{B}$ stop $\rightsquigarrow$ OPTIMAL
  else choose $x_i < 0$, $i \in \mathcal{B}$ (pricing)
- compute max. steplength α (ratiotest)
- if $\alpha = \infty$ stop $\rightsquigarrow$ INFEASIBLE
  else update $\mathcal{B}$, x, y, r, $A_{\mathcal{B}}^{-1}$

Summary

- Discrete and continuous:
  - vertices are uniquely determined by n equalities
  - if optimal: there always exists an optimal vertex solution
  - solution values are computed numerically
- Reduced costs quantify impact of (non-basic) variable on the objective
- efficient hot starts for re-optimization

- Discrete and continuous:
  - vertices are uniquely determined by n equalities
  - if optimal: there always exists an optimal vertex solution
  - solution values are computed numerically
- Reduced costs quantify impact of (non-basic) variable on the objective
- efficient hot starts for re-optimization

Further aspects:

- general bounds: $lb_i \leq x_i \leq ub_i$
- feasible starting basis for simplex ("phase 1"), pricing strategies, linear algebra tricks, ...
- exponential worst-case complexity of simplex vs. performance in practice
- interior point algorithms
- algorithms for specially structured LPs: networks, ...

Integer Programming for Constraint Programmers

Linear program

Objective function:

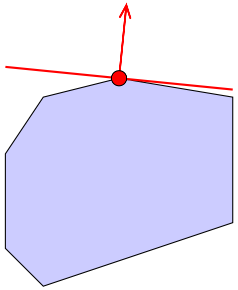
▷ linear function

Feasible set:

▷ described by linear constraints

Variable domains:

▷ real values



$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \in \mathbb{R}^n_{\geq 0} \end{aligned}$$

▷ convex set

▷ “basic” solutions

Integer Program

Objective function:

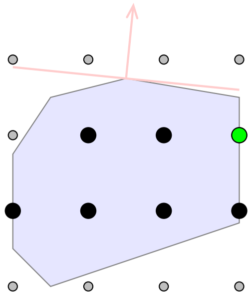
- ▷ linear function

Feasible set:

- ▷ described by linear constraints

Variable domains:

- ▷ integer values



$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax \leq b \\ & x \in \mathbb{Z}_{\geq 0} \end{aligned}$$

- ▷ not even connected
- ▷ $\mathcal{NP}$-hard problem

An incomplete history on integer programming

Cutting plane algorithm

- R. E. Gomory, “Outline of an algorithm for integer solutions to linear programs”. Bull. AMS 64, 1958, pp. 275–278.

Cutting plane algorithm

- R. E. Gomory, “Outline of an algorithm for integer solutions to linear programs”. Bull. AMS 64, 1958, pp. 275–278.

Branch-and-bound

- A. H. Land, A. G. Doig, “An automatic method of solving discrete programming problems”. Econometrica 28, 1960, pp. 497–520
- R. J. Dakin, “A tree-search algorithm for mixed integer programming problems”. The Computer Journal, Volume 8, 1965, pp. 250–255
- J. D. C. Little, K. G. Murty, D. W. Sweeney, C. Karel, “An algorithm for the traveling salesman problem”. Operations Research 11, 1963, pp. 972–989.

Cutting plane algorithm

- R. E. Gomory, "Outline of an algorithm for integer solutions to linear programs". Bull. AMS 64, 1958, pp. 275–278.

Branch-and-bound

- A. H. Land, A. G. Doig, "An automatic method of solving discrete programming problems". Econometrica 28, 1960, pp. 497–520
- R. J. Dakin, "A tree-search algorithm for mixed integer programming problems". The Computer Journal, Volume 8, 1965, pp. 250–255
- J. D. C. Little, K. G. Murty, D. W. Sweeney, C. Karel, "An algorithm for the traveling salesman problem". Operations Research 11, 1963, pp. 972–989.

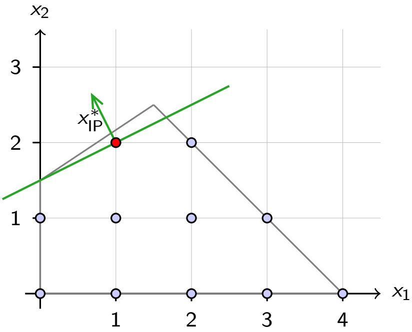
Branch-and-cut

- Grötschel, Jünger, Reinelt (1984, 1985, 1987)
- Padberg, Rinaldi (1991)

General cutting plane method

$$\mathcal{F}_{\text{IP}} := \{x \in \mathbb{Z}^n_+ : Ax \leq b\}$$

$$\mathcal{F}_{\text{LP}} := \{x \in \mathbb{R}^n_+ : Ax \leq b\}$$



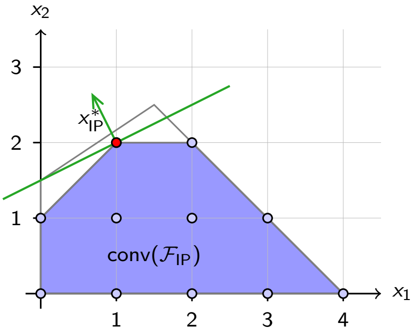
$\min\{c^T x : x \in \mathcal{F}_{\text{IP}}\}$

Observation

- $\text{conv}(\mathcal{F}_{\text{IP}})$ is a polyhedron
- IP could be formulated as LP

Problems with $\text{conv}(\mathcal{F}_{\text{IP}})$:

- linear description not known
- large nr. of constraints needed



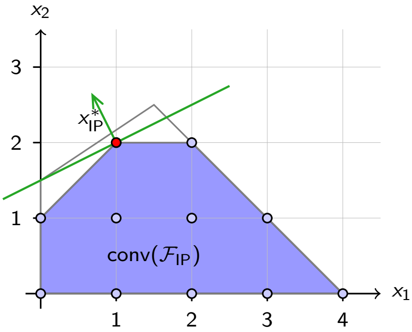
$$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$$

Observation

- ▷ $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ is a polyhedron
- ▷ IP could be formulated as LP

Problems with $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$:

- ▷ linear description not known
- ▷ large nr. of constraints needed



$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$

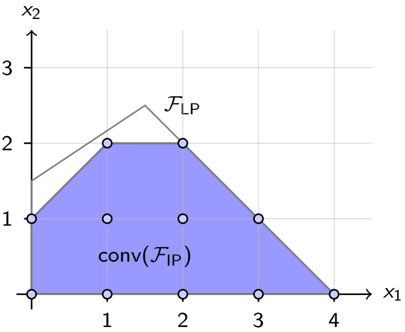
$$\mathcal{F}_{\mathrm{LP}} \quad \supseteq \qquad \mathcal{F} \quad \supseteq \qquad \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$$

$$\min\{c^T x : x \in \mathcal{F}_{\mathrm{LP}}\} \leq \min\{c^T x : x \in \mathcal{F}\} = \min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$$

General cutting plane method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x \in \mathcal{F} \end{aligned}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



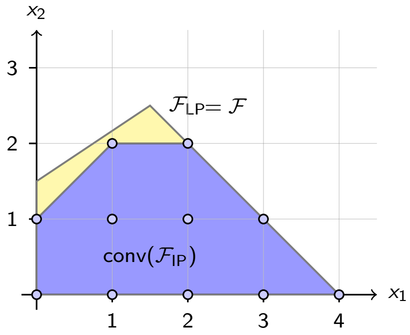
$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



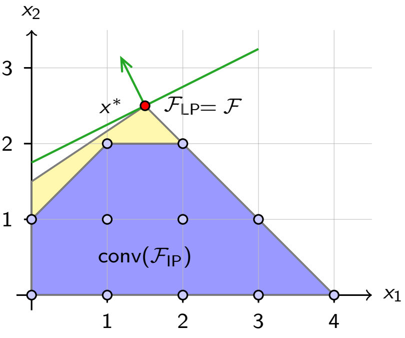
$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



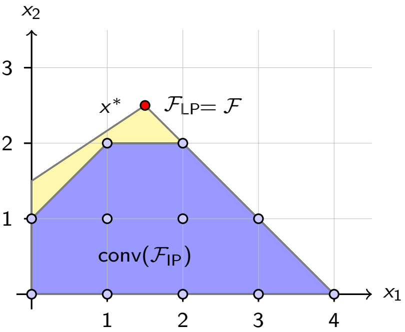
$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{rl} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

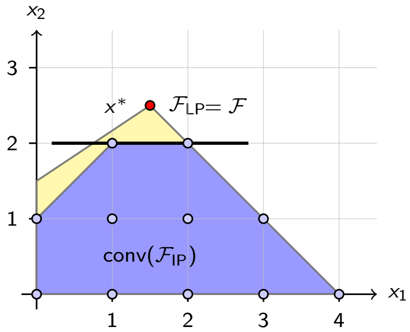
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\mathrm{LP}}$
2. Solve
$$\begin{array}{rl} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\mathrm{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ but
   - violated by x^*.
5. Goto 2.



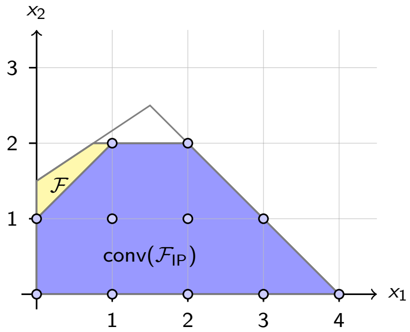
$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\mathrm{LP}}$
2. Solve

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x \in \mathcal{F} \end{aligned}$$

3. If $x^* \in \mathcal{F}_{\mathrm{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$

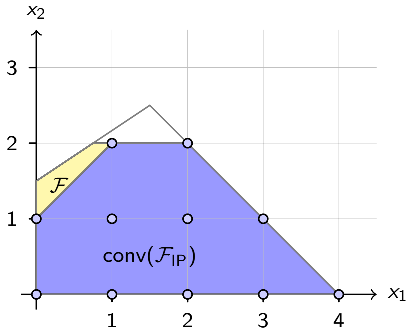
General cutting plane method

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve

$$\begin{array}{rr} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$

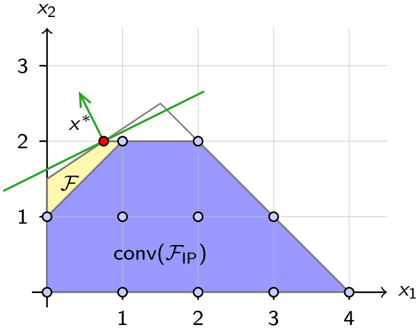
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x \in \mathcal{F} \end{aligned}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

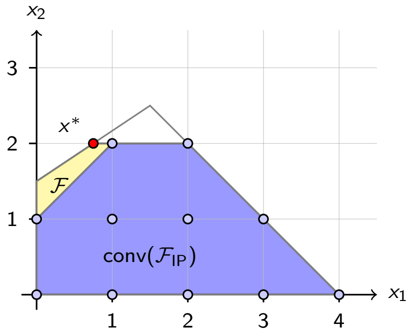


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

- Resolving is cheap since dual feasible (hot start)

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

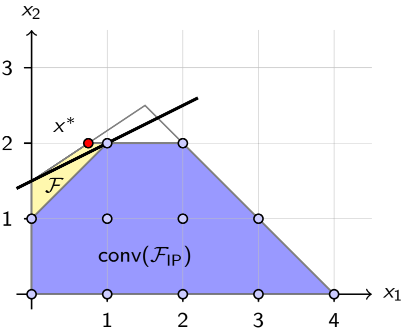
▷ Resolving is cheap since dual feasible (hot start)

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\mathrm{LP}}$
2. Solve

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x \in \mathcal{F} \end{aligned}$$

3. If $x^* \in \mathcal{F}_{\mathrm{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ but
   - violated by x^*.
5. Goto 2.

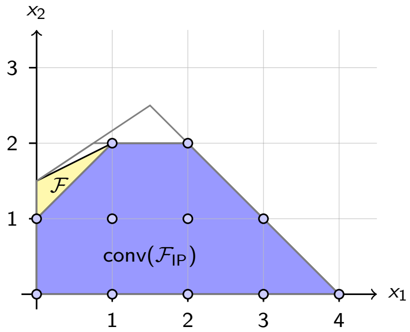


$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$

- Resolving is cheap since dual feasible (hot start)

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

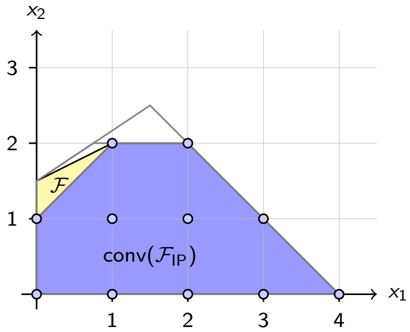


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

- Resolving is cheap since dual feasible (hot start)

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.



$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

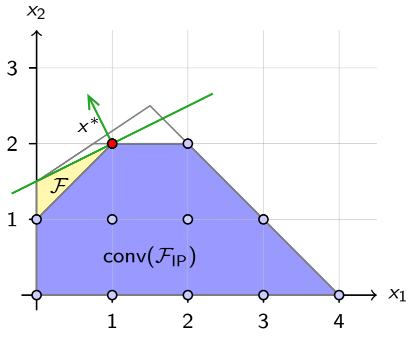
▷ Resolving is cheap since dual feasible (hot start)

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\mathrm{LP}}$
2. Solve

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x \in \mathcal{F} \end{aligned}$$

3. If $x^* \in \mathcal{F}_{\mathrm{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is …
   - valid for $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ but
   - violated by x^*.
5. Goto 2.

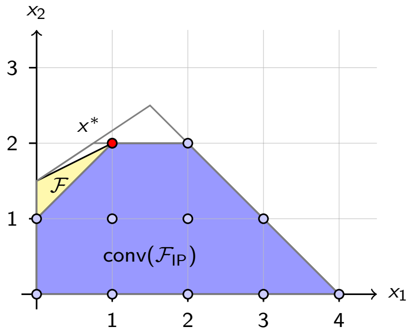


$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$

▷ Resolving is cheap since dual feasible (hot start)

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\text{LP}}$
2. Solve
$$\begin{array}{rl} \min & c^T x \\ \text{s.t.} & x \in \mathcal{F} \end{array}$$
3. If $x^* \in \mathcal{F}_{\text{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\text{conv}(\mathcal{F}_{\text{IP}})$ but
   - violated by x^*.
5. Goto 2.

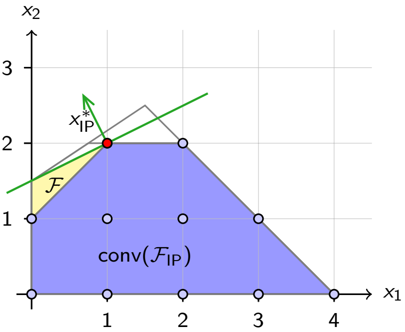


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

▷ Resolving is cheap since dual feasible (hot start)

Algorithm

1. $\mathcal{F} \leftarrow \mathcal{F}_{\mathrm{LP}}$
2. Solve
$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & x \in \mathcal{F} \end{aligned}$$
3. If $x^* \in \mathcal{F}_{\mathrm{IP}}$: Stop
4. Add inequality to $\mathcal{F}$ that is ...
   - valid for $\mathrm{conv}(\mathcal{F}_{\mathrm{IP}})$ but
   - violated by x^*.
5. Goto 2.

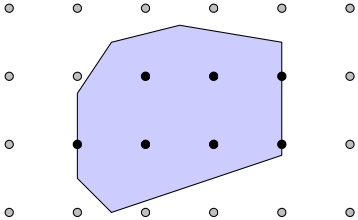


$\min\{c^T x : x \in \mathrm{conv}(\mathcal{F}_{\mathrm{IP}})\}$

▷ Resolving is cheap since dual feasible (hot start)

Steps

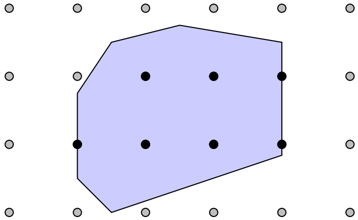
1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching



LP-based branch-and-bound (colorful picture)

Steps

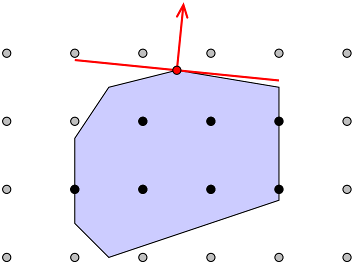
1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching



LP-based branch-and-bound (colorful picture)

Steps

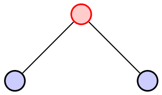
1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

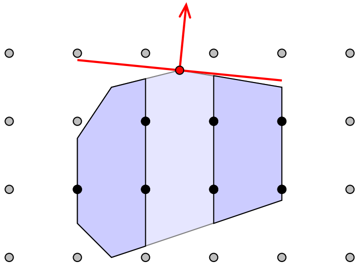


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

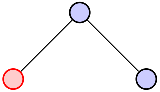


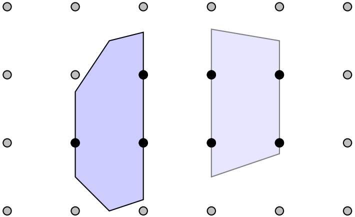


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

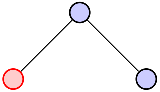


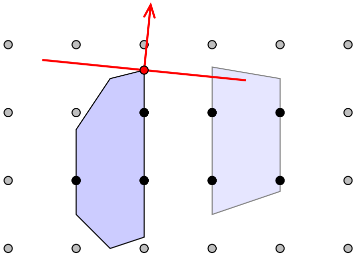


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

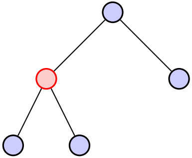


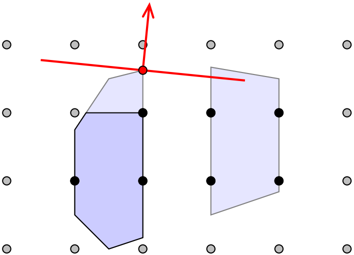


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

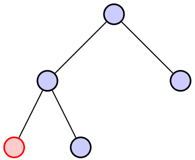


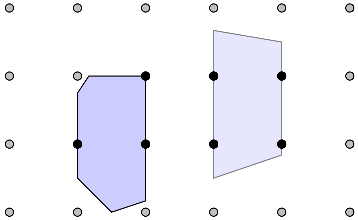


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

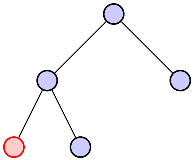


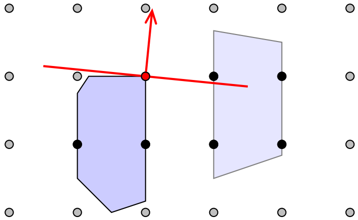


LP-based branch-and-bound (colorful picture)

Steps

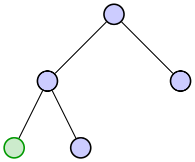
1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

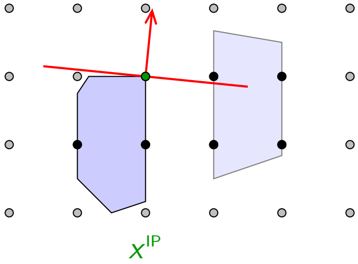




Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

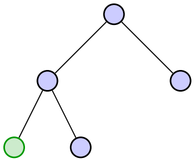


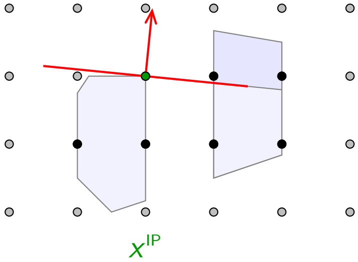


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

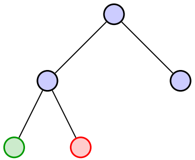


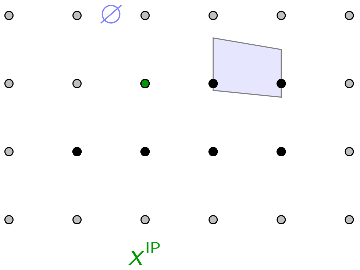


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

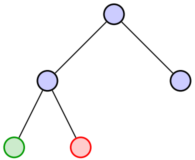


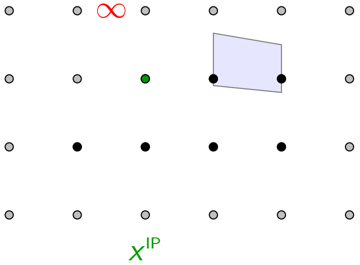


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

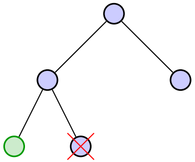


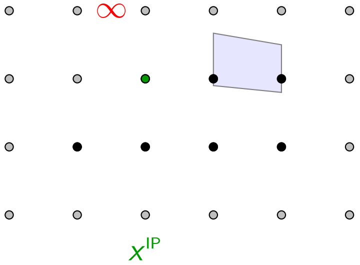


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

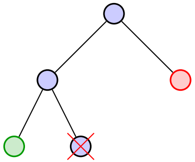


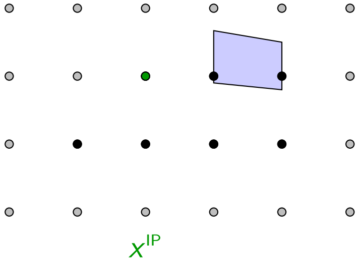


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

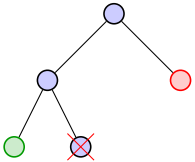


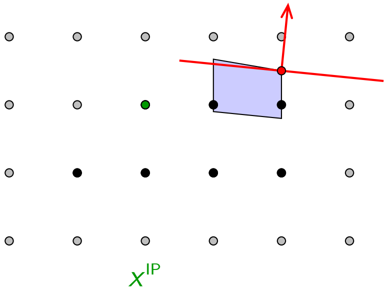


LP-based branch-and-bound (colorful picture)

Steps

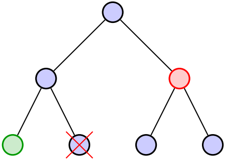
1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

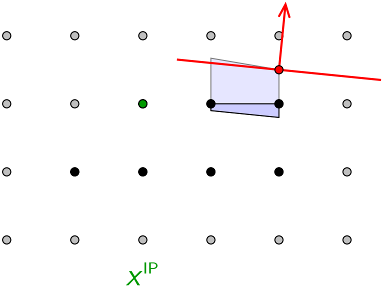




Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

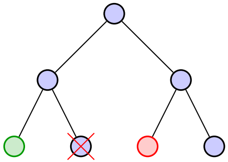


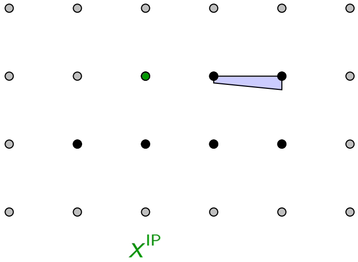


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

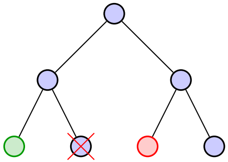


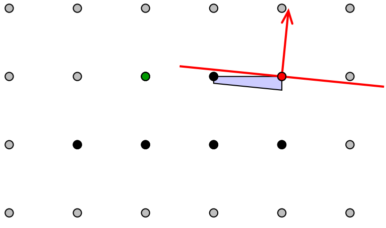


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

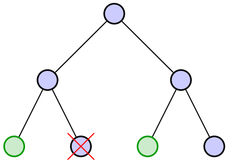


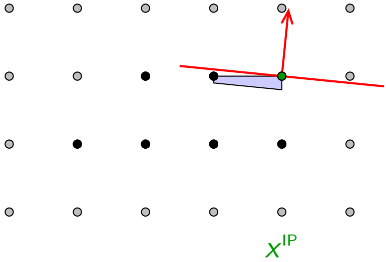


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

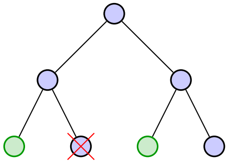


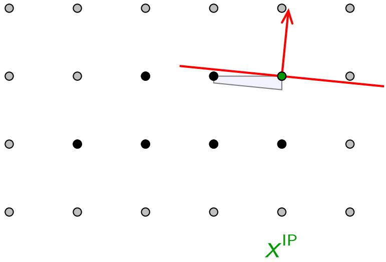


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

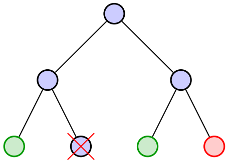


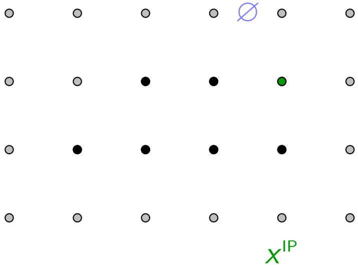


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

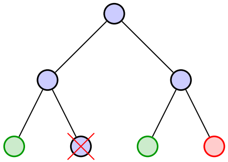


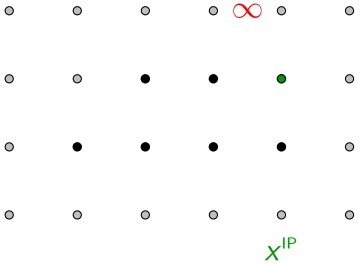


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

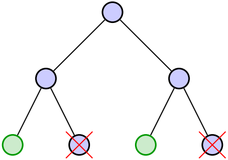


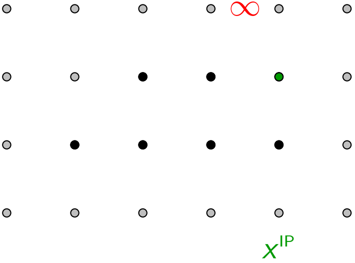


LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

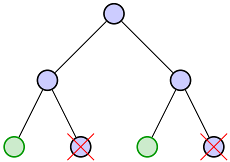


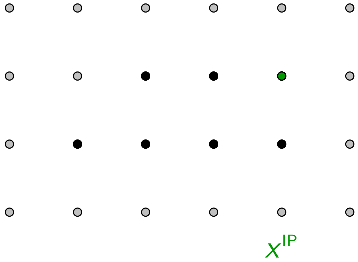


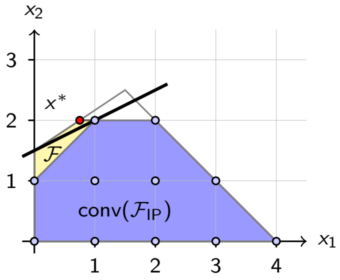
LP-based branch-and-bound (colorful picture)

Steps

1. Abort criterion
2. Node selection
3. Solve relaxation
4. Bounding
5. Feasibility check
6. Branching

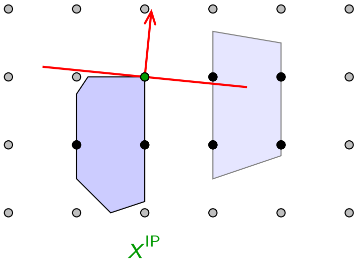




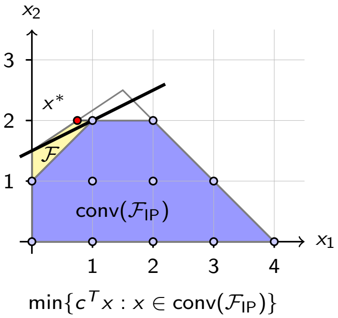


$\min\{c^T x : x \in \text{conv}(\mathcal{F}_{\text{IP}})\}$

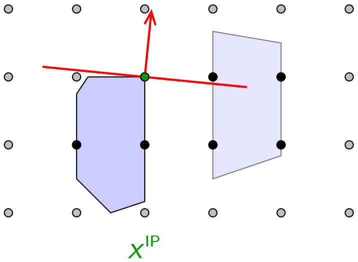
cutting planes



branch-and-bound



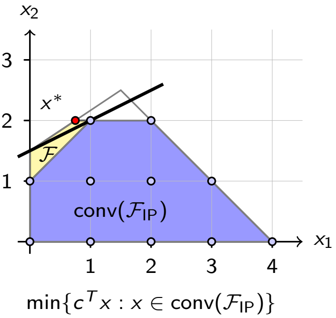
cutting planes



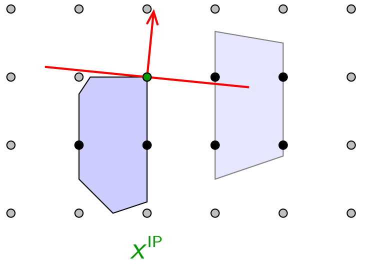
branch-and-bound

- Both approaches solve an initial linear program.

Solving an integer program

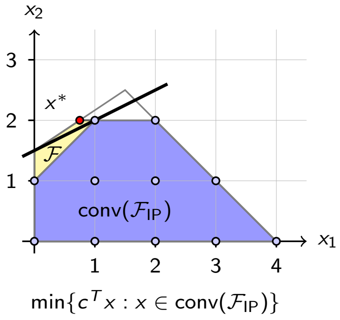


cutting planes

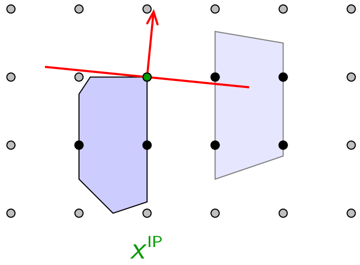


branch-and-bound

- Both approaches solve an initial linear program.
- Classically using simplex algorithm for efficent hot start



cutting planes



branch-and-bound

- Both approaches solve an initial linear program.
- Classically using simplex algorithm for efficent hot start
  - A cutting plane or a bound change is an additional row (linear constraint).

Branch-and-cut $=$ Branch-and-bound $+$ Cutting planes

Branch-and-cut = Branch-and-bound + Cutting planes

- Use branch-and-bound as global solver paradigm.
- Perform limited cutting plane generation within each search node.
  - global versus local cuts

Branch-and-cut = Branch-and-bound + Cutting planes

- Use branch-and-bound as global solver paradigm.
- Perform limited cutting plane generation within each search node.
  - global versus local cuts
- Pure cutting plane generation does not work in practice
  - numerical issues
  - convergence

Branch-and-cut = Branch-and-bound + Cutting planes

- Use branch-and-bound as global solver paradigm.
- Perform limited cutting plane generation within each search node.
  - global versus local cuts

- Pure cutting plane generation does not work in practice
  - numerical issues
  - convergence
- Pure branch-and-bound fails in general
  - exponential search tree

Branch-and-cut = Branch-and-bound + Cutting planes

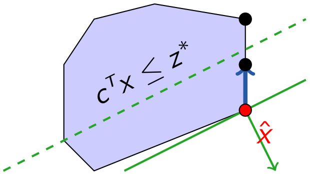
- Use branch-and-bound as global solver paradigm.
- Perform limited cutting plane generation within each search node.
  - global versus local cuts

- Pure cutting plane generation does not work in practice
  - numerical issues
  - convergence
- Pure branch-and-bound fails in general
  - exponential search tree
- Branch-and-cut fails later
  - still exponential search tree
  - but shifts the exponential grow significantly

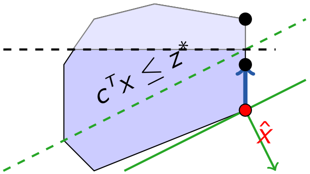
- How can a linear program be solved?
- How can an integer program be solved?
- For what is the linear programming relaxation used within an integer programming solver?

Reduced cost propagation

- z^*: best objective value $\rightarrow$ $c^T x \leq z^*$
- $\hat{x}$: LP optimum
- For variables x_i with reduced cost $r_i \neq 0$
  - variables are not in the basis
  - variables sitting on one of their bounds
  - $r_i > 0 \rightarrow \hat{x}_i = \mathrm{lb}_i$ (lower bound)
  - $r_i < 0 \rightarrow \hat{x}_i = \mathrm{ub}_i$ (upper bound)



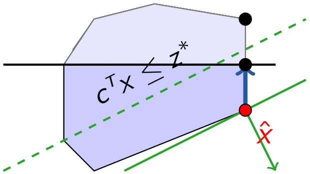
- z^*: best objective value $\rightarrow c^T x \leq z^*$
- $\hat{x}$: LP optimum
- For variables x_i with reduced cost $r_i \neq 0$
  - variables are not in the basis
  - variables sitting on one of their bounds
  - $r_i > 0 \rightarrow \hat{x}_i = \mathrm{lb}_i$ (lower bound)
  - $r_i < 0 \rightarrow \hat{x}_i = \mathrm{ub}_i$ (upper bound)



Case 1 $r_i > 0$:

$$c^T\hat{x} + r_i\,(x_i - \mathrm{lb}_i) \leq z^* \Leftrightarrow x_i \leq \frac{z^* - c^T\hat{x}}{r_i} + \mathrm{lb}_i$$

- z^*: best objective value $\rightarrow$ $c^T x \leq z^*$
- $\hat{x}$: LP optimum
- For variables x_i with reduced cost $r_i \neq 0$
  - variables are not in the basis
  - variables sitting on one of their bounds
  - $r_i > 0 \rightarrow \hat{x}_i = \text{lb}_i$ (lower bound)
  - $r_i < 0 \rightarrow \hat{x}_i = \text{ub}_i$ (upper bound)

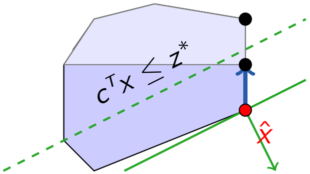


Case 1 $r_i > 0$:

$$c^T \hat{x} + r_i (x_i - \text{lb}_i) \leq z^* \Leftrightarrow x_i \leq \frac{z^* - c^T \hat{x}}{r_i} + \text{lb}_i \Rightarrow x_i \leq \left\lfloor \frac{z^* - c^T \hat{x}}{r_i} + \text{lb}_i \right\rfloor$$

Reduced cost propagation

- z^*: best objective value $\rightarrow$ $c^Tx \leq z^*$
- $\hat{x}$: LP optimum
- For variables x_i with reduced cost $r_i \neq 0$
  - variables are not in the basis
  - variables sitting on one of their bounds
  - $r_i > 0 \rightarrow \hat{x}_i = \mathrm{lb}_i$ (lower bound)
  - $r_i < 0 \rightarrow \hat{x}_i = \mathrm{ub}_i$ (upper bound)



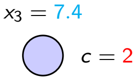
Case 1 $r_i > 0$:

$$c^T\hat{x} + r_i\,(x_i - \mathrm{lb}_i) \leq z^* \Leftrightarrow x_i \leq \frac{z^* - c^T\hat{x}}{r_i} + \mathrm{lb}_i \Rightarrow x_i \leq \left\lfloor \frac{z^* - c^T\hat{x}}{r_i} + \mathrm{lb}_i \right\rfloor$$

Case 2 $r_i < 0$:

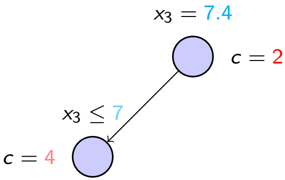
$$c^T\hat{x} + r_i\,(x_i - \mathrm{ub}_i) \leq z^* \Leftrightarrow x_i \geq \frac{z^* - c^T\hat{x}}{r_i} + \mathrm{ub}_i \Rightarrow x_i \geq \left\lceil \frac{z^* - c^T\hat{x}}{r_i} + \mathrm{ub}_i \right\rceil$$

Estimating the objective



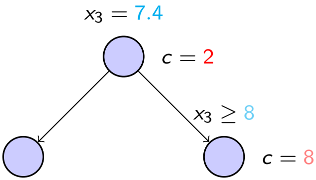
Estimating the objective

- objective gain per unit:
  - $\zeta^{-}(x_3) = \frac{4-2}{7.4-7} = \frac{2}{0.4} = 5$



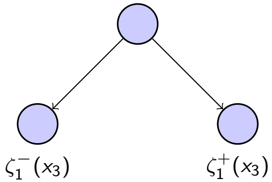
Estimating the objective

- objective gain per unit:
  - $\zeta^+(x_3) = \frac{8-2}{8-7.4} = \frac{6}{0.6} = 10$



Estimating the objective

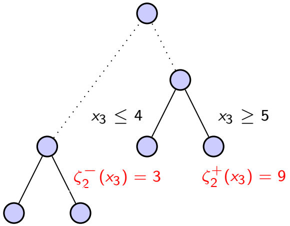
- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$



Estimating the objective

▷ objective gain per unit:

- $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
- other values at other nodes

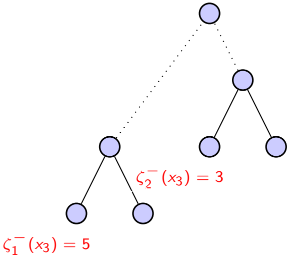


Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- pseudocosts:

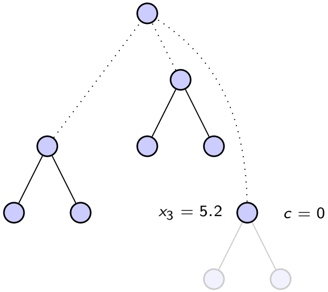
  average objective gain

  $\psi^-(x_3) = \frac{\zeta_1^-(x_3) + \ldots + \zeta_n^-(x_3)}{n} = \frac{5+3}{2} = 4$



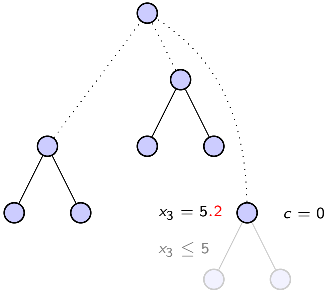
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- estimate increase of objective
  by pseudocosts and fractionality:



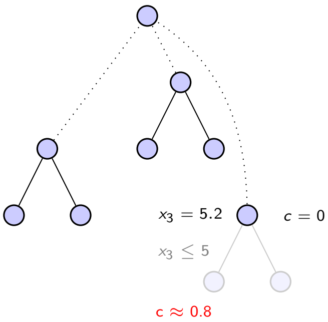
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- estimate increase of objective
  by pseudocosts and fractionality:
  $\psi^-(x_3) \cdot \mathrm{frac}(x_3)$



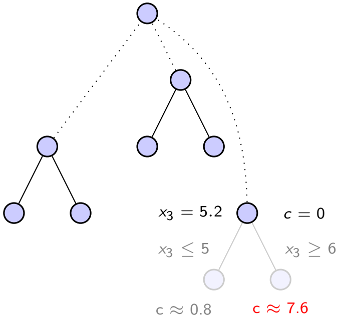
Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- estimate increase of objective
  by pseudocosts and fractionality:
  $\psi^-(x_3) \cdot \mathrm{frac}(x_3) = 4 \cdot 0.2 = 0.8$,



Estimating the objective

- objective gain per unit:
  - $\zeta_1^-(x_3) = 5$, $\zeta_1^+(x_3) = 10$
  - other values at other nodes
- pseudocosts:
  average objective gain
  $\psi^-(x_3) = 4$, $\psi^+(x_3) = 9.5$
- estimate increase of objective
  by pseudocosts and fractionality:
  $\psi^-(x_3) \cdot \mathrm{frac}(x_3) = 4 \cdot 0.2 = 0.8$,
  and $\psi^+(x_3)(1 - \mathrm{frac}(x_3)) = 7.6$

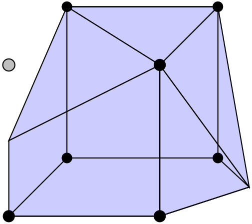


RENS – Relaxation Enforced Neighborhood Search

Idea: Search the vicinity of a relaxation solution

Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i \quad$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables
   $x_i \in \{\lfloor \bar{x}_i \rfloor; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP;

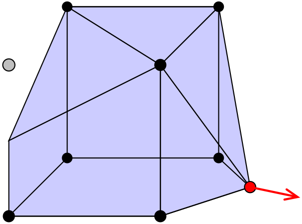


RENS – Relaxation Enforced Neighborhood Search

Idea: Search the vicinity of a relaxation solution

Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables
   $x_i \in \{\lfloor \bar{x}_i \rfloor; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP;

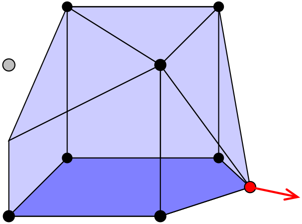


RENS – Relaxation Enforced Neighborhood Search

Idea: Search the vicinity of a relaxation solution

Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables
   $x_i \in \{\lfloor \bar{x}_i \rfloor; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP;

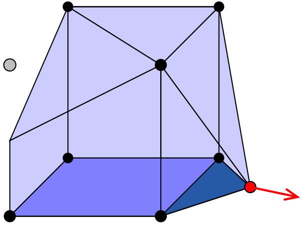


RENS – Relaxation Enforced Neighborhood Search

Idea: Search the vicinity of a relaxation solution

Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables
   $x_i \in \{\lfloor \bar{x}_i \rfloor ; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP;

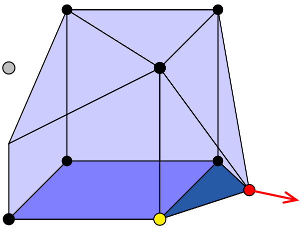


RENS – Relaxation Enforced Neighborhood Search

Idea: Search the vicinity of a relaxation solution

Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i \quad$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables
   $x_i \in \{\lfloor \bar{x}_i \rfloor; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP;

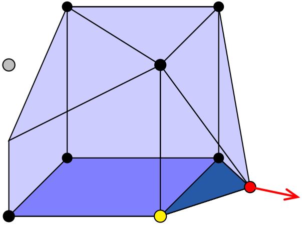


RENS – Relaxation Enforced Neighborhood Search

Idea: Search the vicinity of a relaxation solution

Algorithm

1. $\bar{x} \leftarrow$ LP optimum;
2. Fix all integral variables:
   $x_i := \bar{x}_i \quad$ for all $i : \bar{x}_i \in \mathbb{Z}$;
3. Reduce domain of fractional variables
   $x_i \in \{\lfloor \bar{x}_i \rfloor; \lceil \bar{x}_i \rceil\}$;
4. Solve the resulting sub-MIP;



Crucial point: Does not need a feasible start solution

Integer Programming for Constraint Programmers

Linear relaxation

- gives a global view
- provides a proven dual bound for the original problem
  - quality guarantee
- can be used for more than getting a dual bound
  - propagation, branching, primal heuristic, ...

Linear relaxation

- gives a global view
- provides a proven dual bound for the original problem
  - quality guarantee
- can be used for more than getting a dual bound
  - propagation, branching, primal heuristic, ...

Additional remarks

- a linear relaxation does not have to represent all constraints
- numeric issues can arise due to continuous optimization
  - in general the numerics can be controlled
  - there exist critical instances
  - see also exact interger programming

Linear relaxation

- gives a global view
- provides a proven dual bound for the original problem
  - quality guarantee
- can be used for more than getting a dual bound
  - propagation, branching, primal heuristic, ...

Additional remarks

- a linear relaxation does not have to represent all constraints
- numeric issues can arise due to continuous optimization
  - in general the numerics can be controlled
  - there exist critical instances
  - see also exact interger programming

As in CP, the chosen model has a huge impact on the performance of a solver

An incomplete list of IP Solvers

Non-commercial solvers

- CBC (IBM) `https://projects.coin-or.org/Cbc`
- GLPK `http://www.gnu.org/s/glpk/`
- LPSOLVE `http://lpsolve.sourceforge.net/`
- SCIP `http://scip.zib.de`
- SYMPHONY `https://projects.coin-or.org/SYMPHONY`

Commercial solvers

- CPLEX (IBM) `http://www.cplex.com`
- GUROBI `http://www.gurobi.com`
- MOPS `http://www.mops-optimizer.com`
- MOSEK `http://www.mosek.com`
- XPRESS (Fico) `http://www.fico.com`

Integer Programming for Constraint Programmers

Questions

Tutorial

Integer Programming for Constraint Programmers

Ambros Gleixner and Stefan Heinz

Zuse Institute Berlin (ZIB)

Chris Beck, Timo Berthold, and Kati Wolter

DFG Research Center MATHEON
Mathematics for key technologies