

***A generic view at
the **Dantzig-Wolfe decomposition** approach
in Mixed Integer Programming:
paving the way for a generic code***

François Vanderbeck

[www//math.u-bordeaux.fr/~fv](http://www.math.u-bordeaux.fr/~fv)

Laboratoire de Mathématique Appliquées de Bordeaux (MAB)

Université Bordeaux 1

France

Motivation

- ⑥ Use MIP solver v.s. **compete** against
Xpress, Cplex, OSL
→ push back their limitations

Motivation

- ⑥ Use MIP solver v.s. **compete** against
Xpress, Cplex, OSL
→ push back their limitations
- ⑥ DW decomposition = **well suited tool** to defer
combinatorial explosion

Motivation

- ⑥ Use MIP solver v.s. **compete** against
Xpress, Cplex, OSL
→ push back their limitations
- ⑥ DW decomposition = **well suited tool** to defer
combinatorial explosion
- ⑥ DW decomposition **not use** to its full power
(f.i.: Lagrangian decomposition)

Motivation

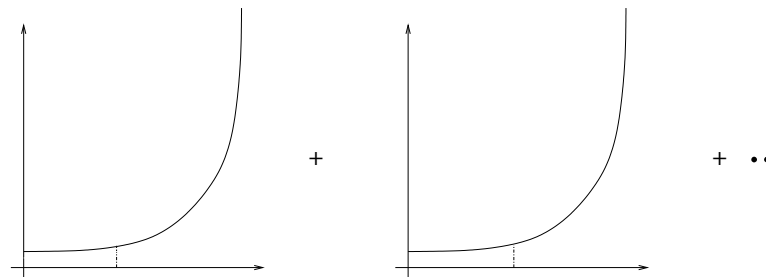
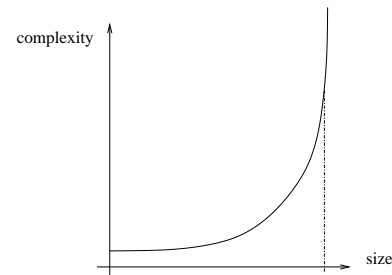
- ⑥ Use MIP solver v.s. **compete** against
Xpress, Cplex, OSL
→ push back their limitations
- ⑥ DW decomposition = **well suited tool** to defer
combinatorial explosion
- ⑥ DW decomposition **not use** to its full power
(f.i.: Lagrangian decomposition)
- ⑥ **Need** for a generic branch-and-price **code**:
Minto (Savelsbergh), Abacus (Thienel), . . .

Motivation

- ⑥ Use MIP solver v.s. **compete** against
Xpress, Cplex, OSL
→ push back their limitations
- ⑥ DW decomposition = **well suited tool** to defer
combinatorial explosion
- ⑥ DW decomposition **not use** to its full power
(f.i.: Lagrangian decomposition)
- ⑥ **Need** for a generic branch-and-price **code**:
Minto (Savelsbergh), Abacus (Thienel), . . .
- ⑥ Generic code ← **generic** understanding of
important ingredients for DW decomp.

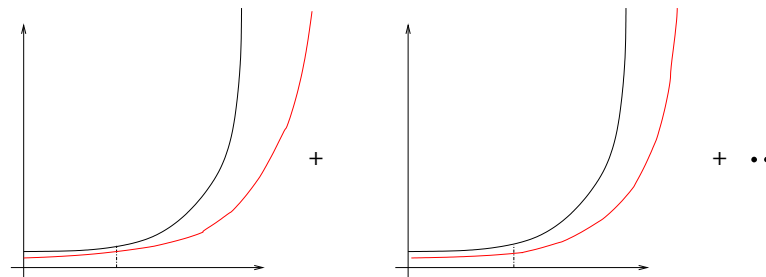
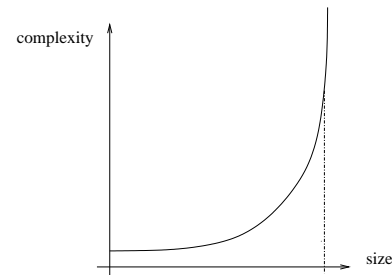
Decomposition: Why

⑥ Divide and Conquer



Decomposition: Why

⑥ Divide and Conquer



⑥ Exploit the structure

Decomposition: When

$$\begin{aligned} Z^{MIP} = \min \quad & c(x, y) \\ & A(x, y) \geq a \\ & B(x, y) \geq b \\ & x \geq 0 \\ & y \in \mathbf{N}^p \end{aligned}$$

Decomposition: When

$$Z^{MIP} = \min c(x, y)$$

$$A(x, y) \geq a$$

$$B(x, y) \geq b$$

$$x \geq 0$$

$$y \in \mathbf{N}^p$$

5 → Difficult Constraints

Decomposition: When

$$Z^{MIP} = \min c(x, y)$$

$$A(x, y) \geq a$$

$$B(x, y) \geq b$$

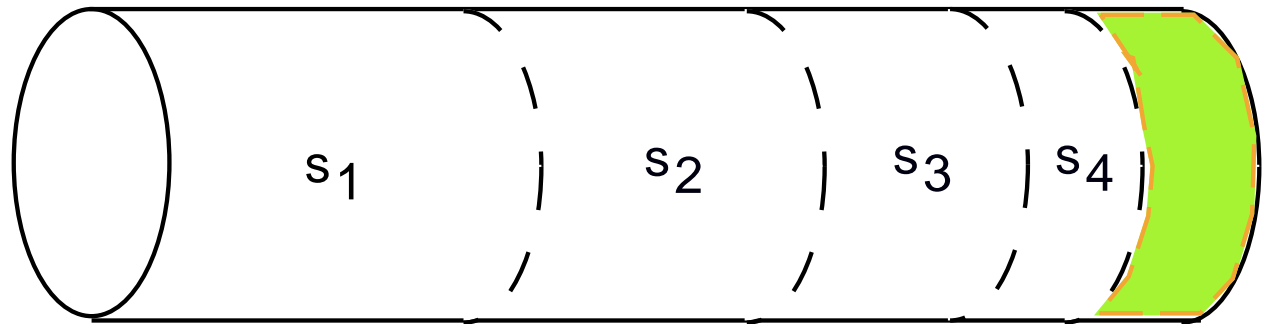
$$x \geq 0$$

$$y \in \mathbf{N}^p$$

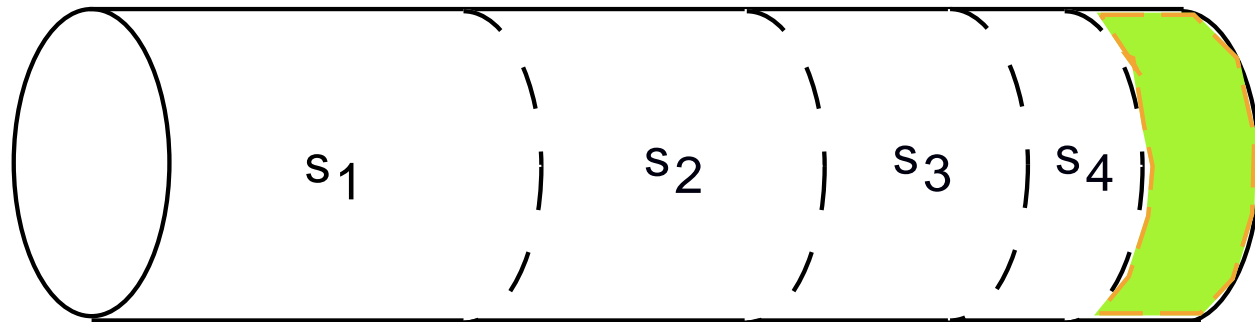
⑥ Difficult Constraints

⑤ Linking Constraints

Example: Cutting Stock Problem



Example: Cutting Stock Problem



$$\min \sum_k y_k$$

$$\sum_k x_{i,k} \geq d_i \quad \forall i$$

$$\sum_i s_i x_{i,k} \leq y_k \quad \forall k$$

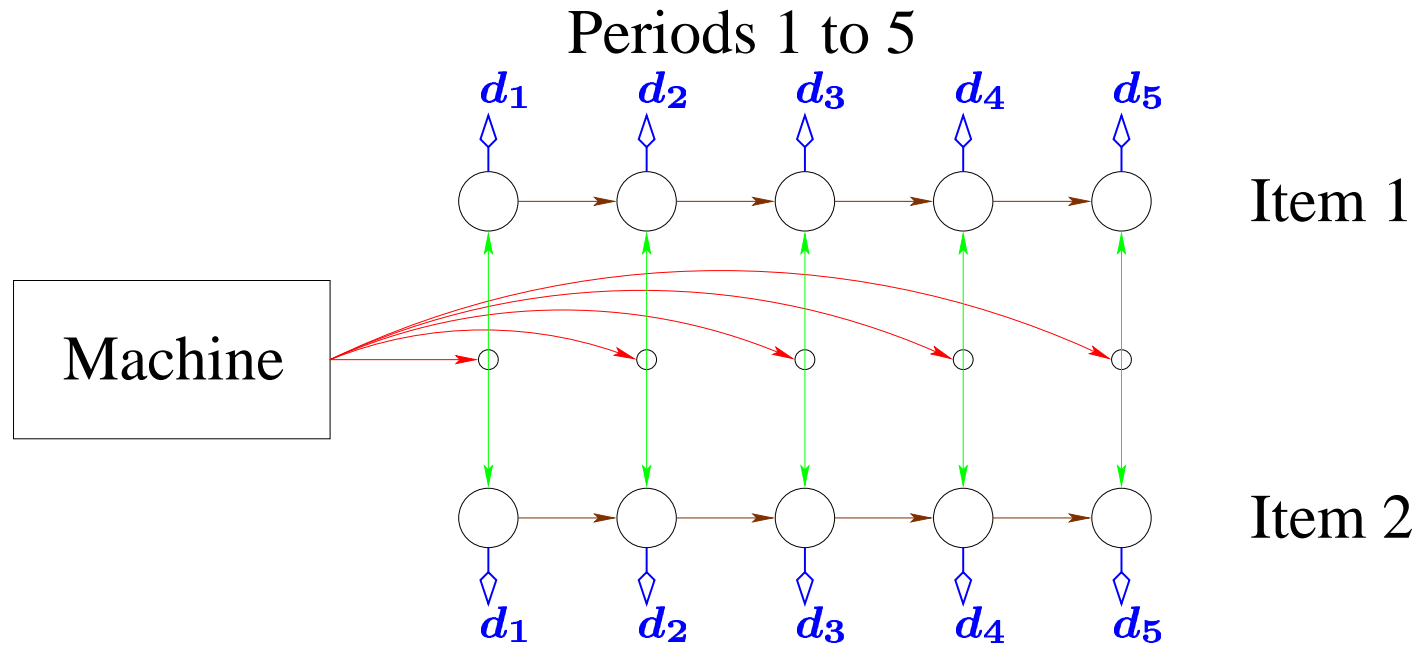
$$(x, y) \in \mathbb{N}^{I+K}$$

Decomposition: When

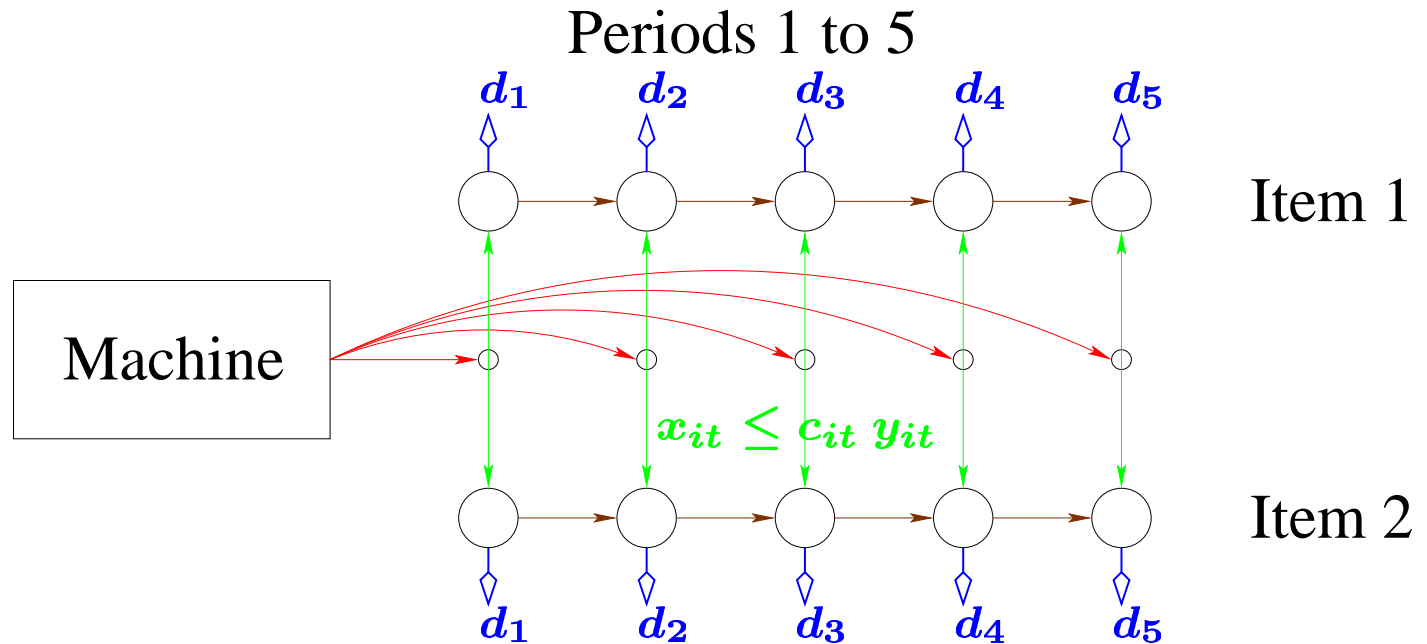
$$\begin{aligned} Z^{MIP} = \min \quad & c(x, y) \\ & A(x, y) \geq a \\ & B(x, y) \geq b \\ & x \geq 0 \\ & y \in \mathbf{N}^p \end{aligned}$$

- ⑥ Difficult Constraints
- ⑥ Linking Constraints
- ⑥ Multiple Sub-Systems (variable splitting)

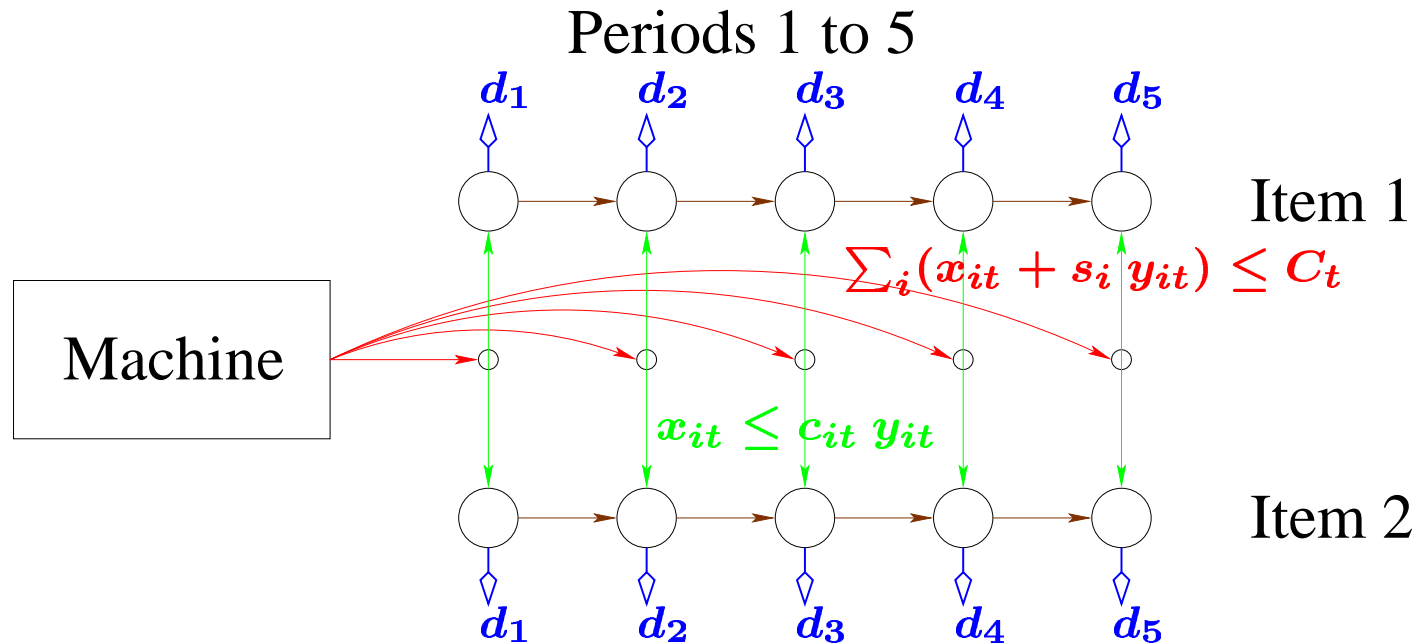
Example: Multi-Item Lot-Sizing



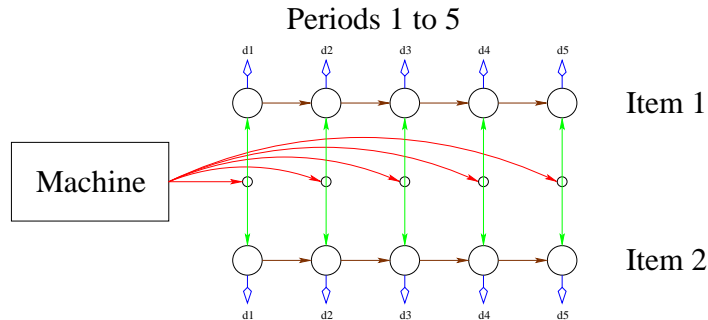
Example: Multi-Item Lot-Sizing



Example: Multi-Item Lot-Sizing



Example: Multi-Item Lot-Sizing



$$\min \sum_{i,t} \left(\frac{p_{it}}{2} x_{it}^A + \frac{f_{it}}{2} y_{it}^A \right) + \sum_{i,t} \left(\frac{p_{it}}{2} x_{it}^B + \frac{f_{it}}{2} y_{it}^B \right)$$

$$x_{it}^A \geq x_{it}^B \quad \forall i, t$$

$$y_{it}^A \geq y_{it}^B \quad \forall i, t$$

$$\sum_i (x_{it}^A + s_i y_{it}^A) \leq C_t \quad \forall t$$

$$\sum_{\tau=1}^t x_{i\tau}^B \geq d_{i1t} \quad \forall i, t$$

$$x_{it}^A \leq c_{it} y_{it}^A \quad \forall i, t$$

$$x_{it}^B \leq c_{it} y_{it}^B \quad \forall i, t$$

$$x_{it}^A \geq 0, y_{it}^A \in \{0, 1\} \quad \forall i, t$$

$$x_{it}^B \geq 0, y_{it}^B \in \{0, 1\} \quad \forall i, t$$

Decomposition: When

$$\begin{aligned} Z^{MIP} = \min \quad & c(x, y) \\ & A(x, y) \geq a \\ & B(x, y) \geq b \\ & x \geq 0 \\ & y \in \mathbf{N}^p \end{aligned}$$

- ⑥ Difficult Constraints
- ⑥ Linking Constraints
- ⑥ Multiple Sub-Systems (variable splitting)

Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

⑥ Lagrangian relaxation (Lagr. Dual)

$$L(\pi) = \min\{c(x, y) + \pi(a - A(x, y)) : B(x, y) \geq b, y \in \mathbb{N}^p\}$$

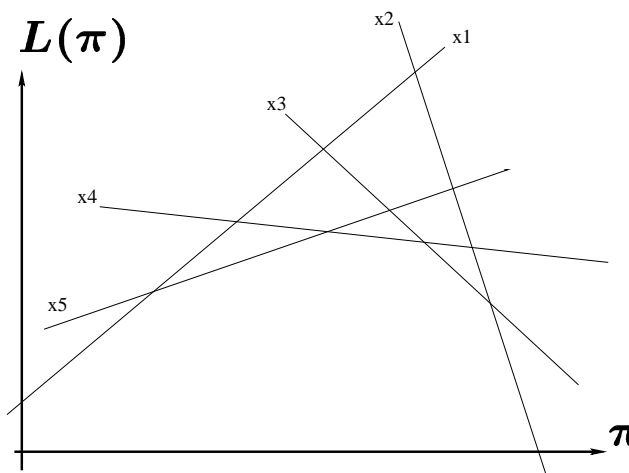
Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

⑥ Lagrangian relaxation (Lagr. Dual)

$$L(\pi) = \min\{c(x, y) + \pi(a - A(x, y)) : B(x, y) \geq b, y \in \mathbb{N}^p\}$$

$$LD = \max_{\pi} L(\pi)$$



Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

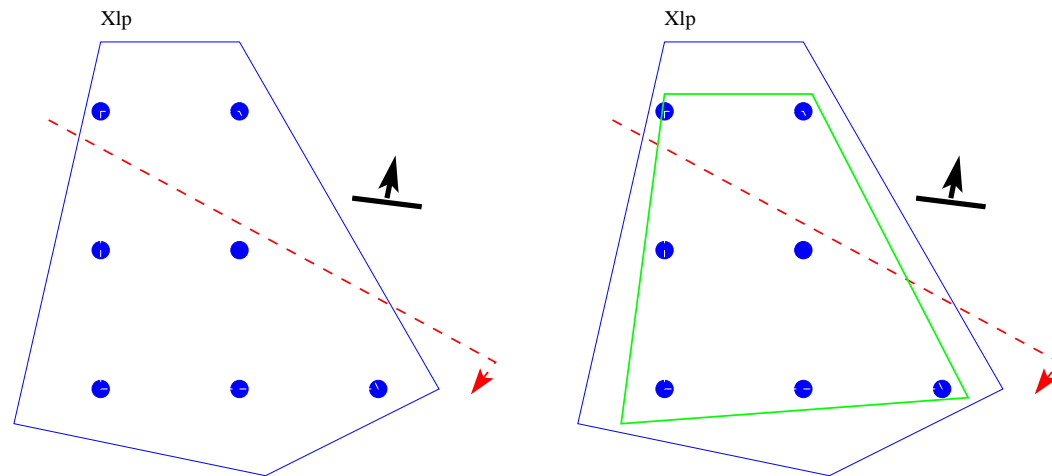
- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)

Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)

$$\{B(x, y) \geq b, (x, y) \geq 0\} \longrightarrow \{B(x, y) \geq b, \gamma(x, y) \geq \gamma_0\}$$



Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)
- ⑥ Reformulation (Variable Redefinition)

Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)
- ⑥ Reformulation (Variable Redefinition)

$$\{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}^p\} \longrightarrow \{G(w, z) \geq g, z \in \mathbb{N}\}$$

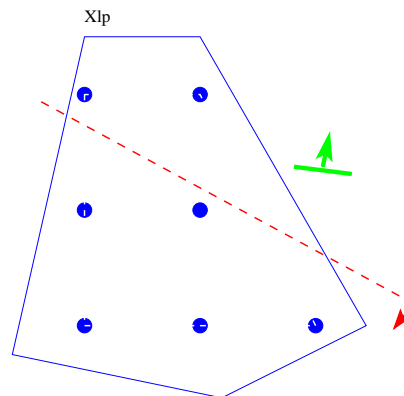
Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)
- ⑥ Reformulation (Variable Redefinition)

Best Dual Bound:

$$\equiv \min\{c(x, y) : A(x, y) \geq a, (x, y) \in \text{conv}(\{B(x, y) \geq b, y \in \mathbb{N}^p\})\}$$



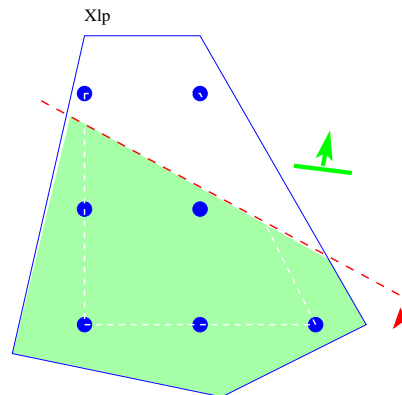
Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)
- ⑥ Reformulation (Variable Redefinition)

Best Dual Bound:

$$\equiv \min\{c(x, y) : A(x, y) \geq a, (x, y) \in \text{conv}(\{B(x, y) \geq b, y \in \mathbb{N}^p\})\}$$



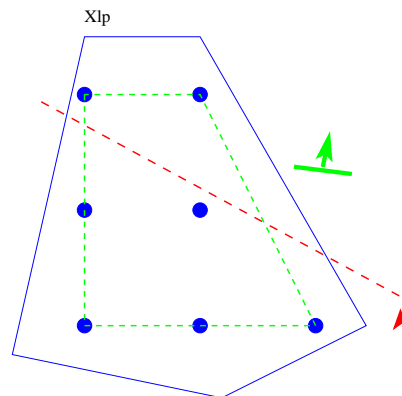
Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)
- ⑥ Reformulation (Variable Redefinition)

Best Dual Bound:

$$\equiv \min\{c(x, y) : A(x, y) \geq a, (x, y) \in \text{conv}(\{B(x, y) \geq b, y \in \mathbb{N}^p\})\}$$



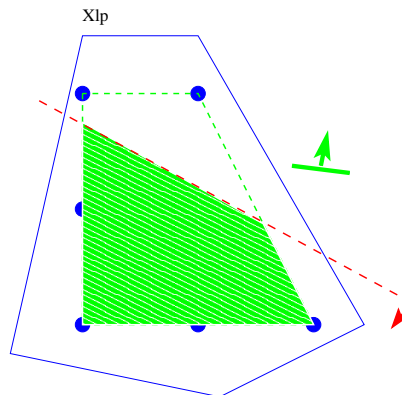
Decomposition: How

$$\min\{c(x, y) : \underbrace{A(x, y) \geq a}_{\text{difficult}}, \underbrace{B(x, y) \geq b, x \geq 0, y \in \mathbb{N}}_{\text{nice}}\}$$

- ⑥ Lagrangian relaxation (Lagr. Dual)
- ⑥ Cut Generation (Separation Sub-Problem)
- ⑥ Reformulation (Variable Redefinition)

Best Dual Bound:

$$\equiv \min\{c(x, y) : A(x, y) \geq a, (x, y) \in \text{conv}(\{B(x, y) \geq b, y \in \mathbb{N}^p\})\}$$



Dantzig-Wolfe Decomposition

reformulation whose LP value achieves the Lagrangian dual b

Dantzig-Wolfe Decomposition

reformulation whose LP value achieves the Lagrangian dual bound

$$X^B = \{(x, y) \in \mathbb{R}_+^n \times \mathbb{N}^p : B(x, y) \geq b\}$$

G^B is a *finite generating set* for X^B

$$X^B \equiv \left\{ (x, y) = \sum_{g \in G^B} g \lambda_g : \sum_{g \in G^B} \lambda_g = 1, \text{ integer restr.} \right\}$$

Dantzig-Wolfe Decomposition

reformulation whose LP value achieves the Lagrangian dual bound

$$X^B = \{(x, y) \in \mathbb{R}_+^n \times \mathbb{N}^p : B(x, y) \geq b\}$$

G^B is a *finite generating set* for X^B

$$X^B \equiv \underbrace{\left\{ (x, y) = \sum_{g \in G^B} g \lambda_g \right\}}_{\text{variable change}} : \underbrace{\left\{ \sum_{g \in G^B} \lambda_g = 1, \text{ integer restr.} \right\}}_{\lambda \in R^B}$$

Re-formulation [MASTER]:

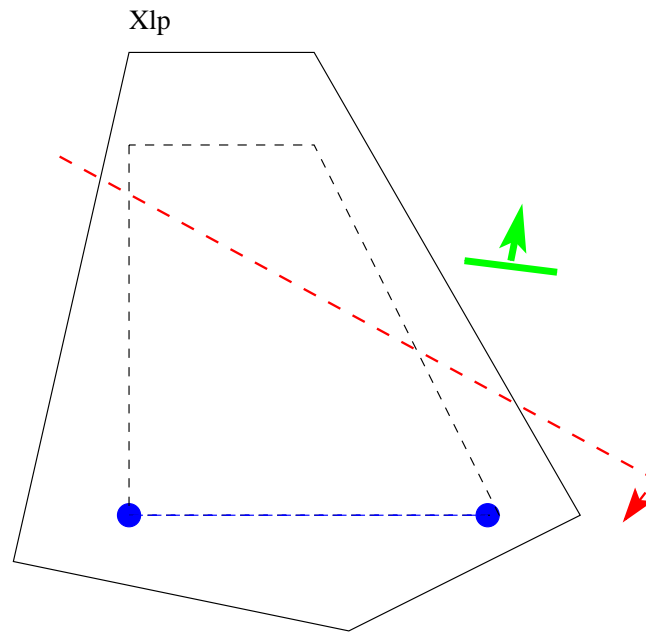
$$\begin{aligned} \min \quad & \sum_{g \in G^B} (c g) \lambda_g \\ & \sum_{g \in G^B} (A g) \lambda_g \geq a \quad (\pi) \\ & \lambda \in R^B \end{aligned}$$

Solving the Master LP

- ⑥ Column Generation: **graphically**

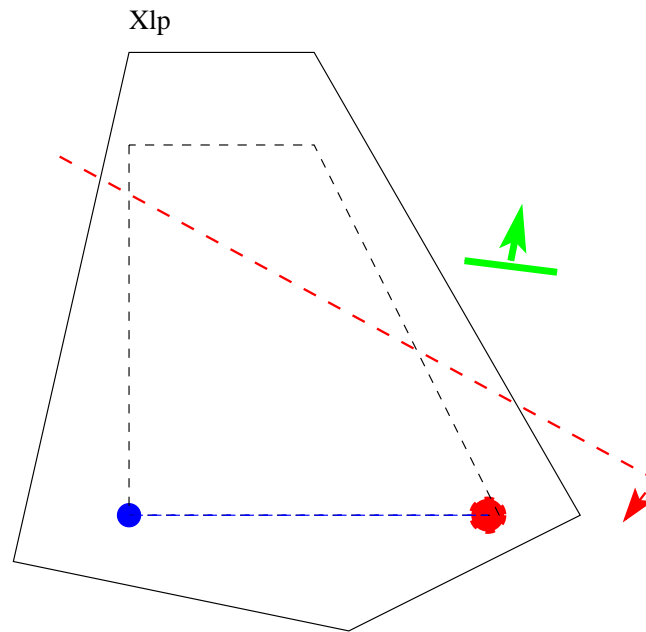
Solving the Master LP

6 Column Generation: **graphically**



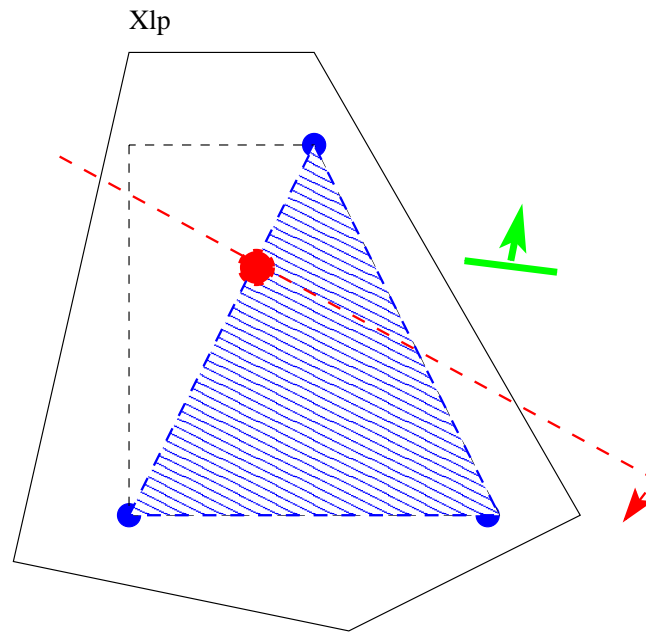
Solving the Master LP

⑥ Column Generation: **graphically**



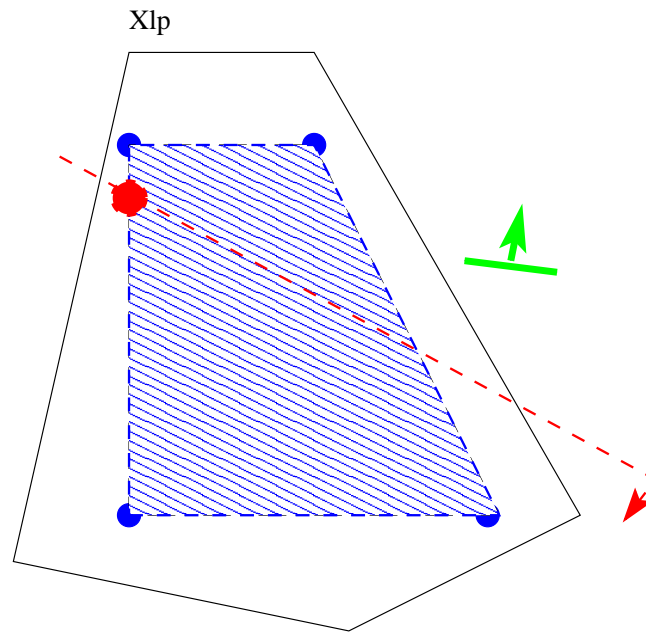
Solving the Master LP

⑥ Column Generation: **graphically**



Solving the Master LP

⑥ Column Generation: **graphically**



Solving the Master LP

6 Column Generation

MASTER:

$$\begin{aligned} \min \quad & \sum_{g \in G^B} (c \ g) \lambda_g \\ & \text{restricted} \\ \sum_{g \in G^B} (A \ g) \lambda_g & \geq a \quad (\pi) \\ & \text{restricted} \\ \lambda & \in R_{LP}^B \end{aligned}$$

SUB-PROBLEM: Check optimality: $\min\{(c - \pi A) g : g \in G^B\} < 0$?

$$\min\{(c - \pi A) (x, y) : B(x, y) \geq b, x \geq 0, y \in \mathbb{N}^p\} .$$

Solving the Master LP

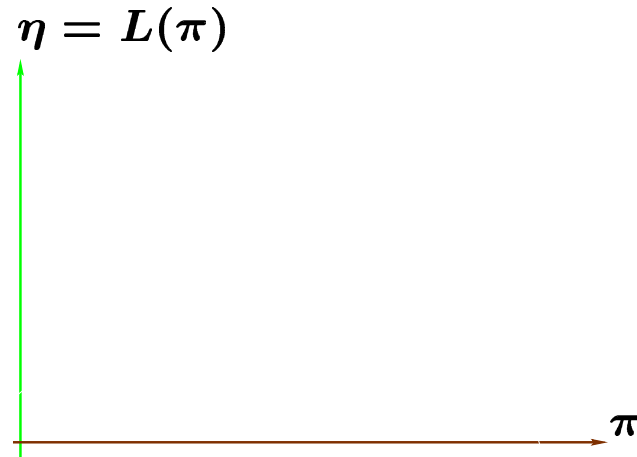
- ⑥ Column Generation = Kelley (in the dual space)
DUAL MASTER:

$$\theta = \max_{(\pi, \eta)} \eta$$
$$\eta + (A g - a) \pi \leq c g \quad g \in G^B$$

Solving the Master LP

- ⑥ Column Generation = Kelley (in the dual space)
DUAL MASTER:

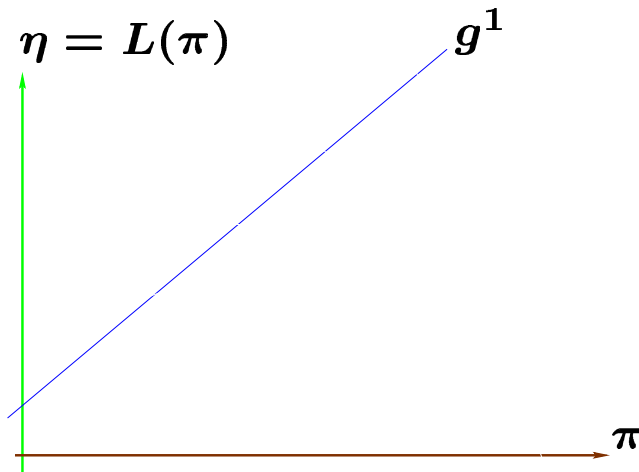
$$\theta = \max_{(\pi, \eta)} \eta$$
$$\eta + (A g - a) \pi \leq c g \quad g \in G^B$$



Solving the Master LP

- ⑥ Column Generation = Kelley (in the dual space)
DUAL MASTER:

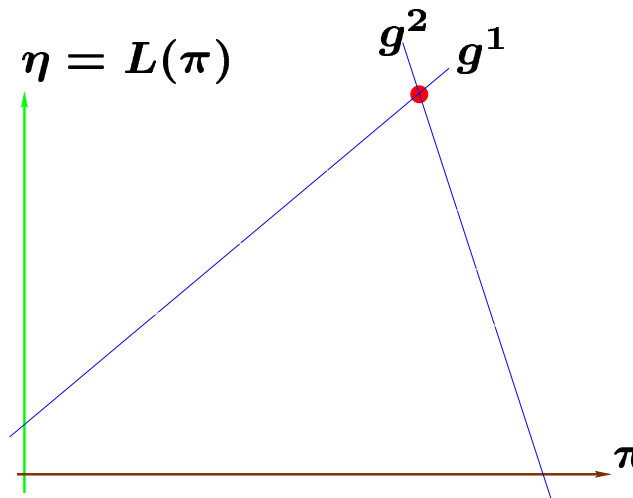
$$\theta = \max_{(\pi, \eta)} \eta$$
$$\eta + (A g - a) \pi \leq c g \quad g \in G^B$$



Solving the Master LP

- ⑥ Column Generation = Kelley (in the dual space)
DUAL MASTER:

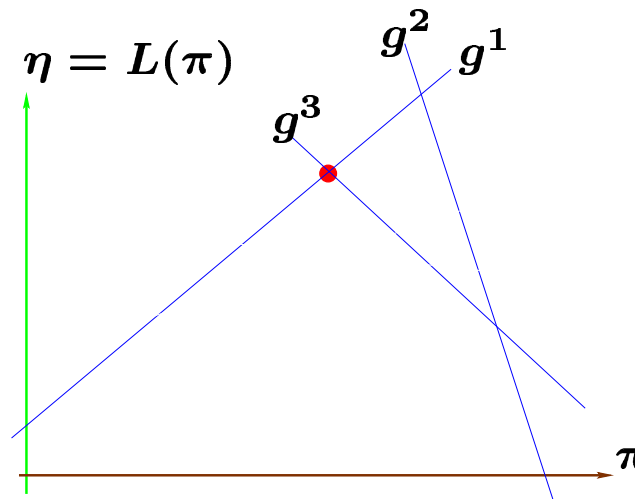
$$\theta = \max_{(\pi, \eta)} \eta$$
$$\eta + (A g - a) \pi \leq c g \quad g \in G^B$$



Solving the Master LP

- ⑥ Column Generation = Kelley (in the dual space)
DUAL MASTER:

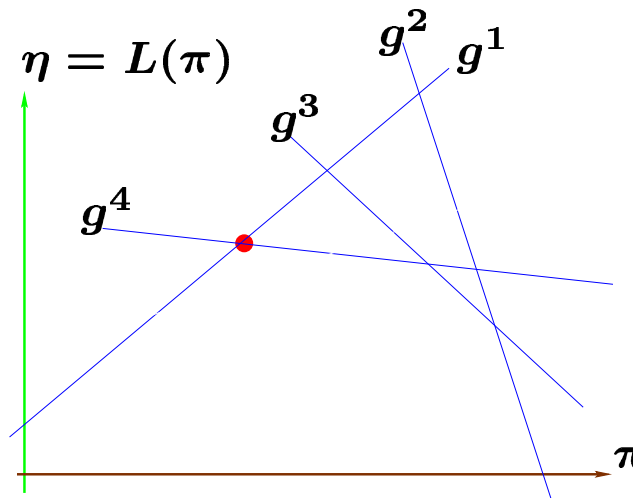
$$\theta = \max_{(\pi, \eta)} \eta$$
$$\eta + (A g - a) \pi \leq c g \quad g \in G^B$$



Solving the Master LP

- ⑥ Column Generation = Kelley (in the dual space)
DUAL MASTER:

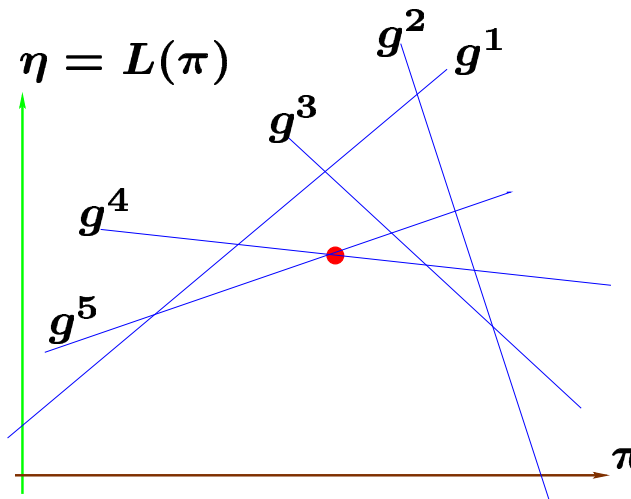
$$\theta = \max_{(\pi, \eta)} \eta$$
$$\eta + (A g - a) \pi \leq c g \quad g \in G^B$$



Solving the Master LP

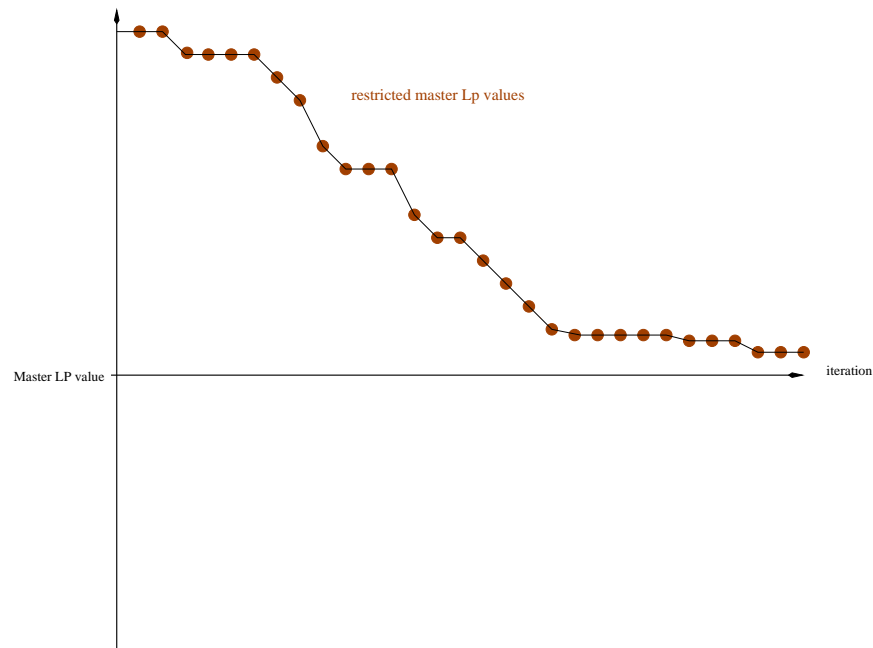
- ⑥ Column Generation = Kelley (in the dual space)
DUAL MASTER:

$$\theta = \max_{(\pi, \eta)} \eta$$
$$\eta + (A g - a) \pi \leq c g \quad g \in G^B$$



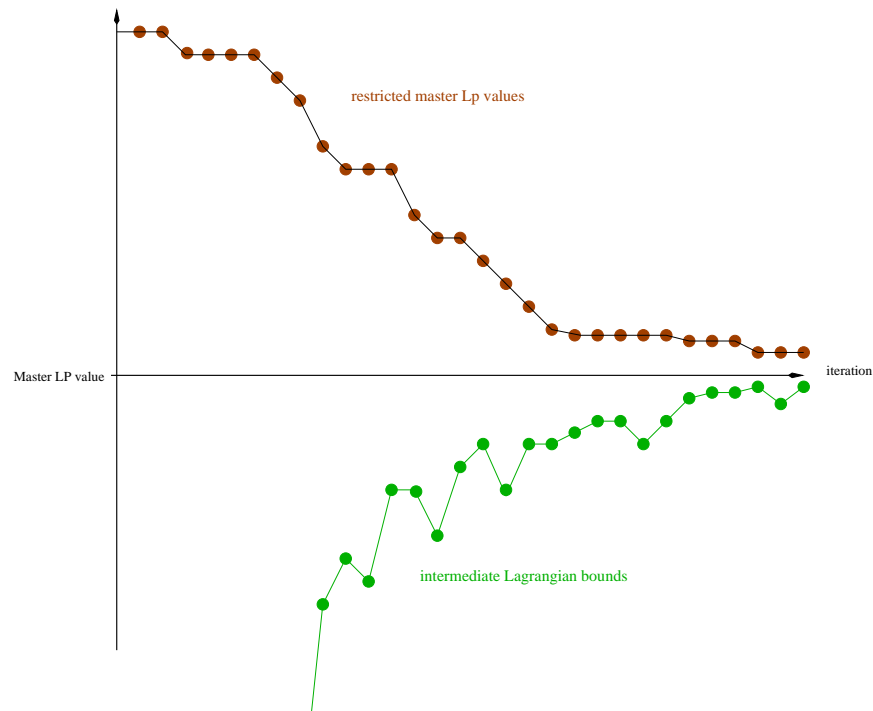
Solving the Master LP

6 Col. Gen. / Kelley: the issue of convergence



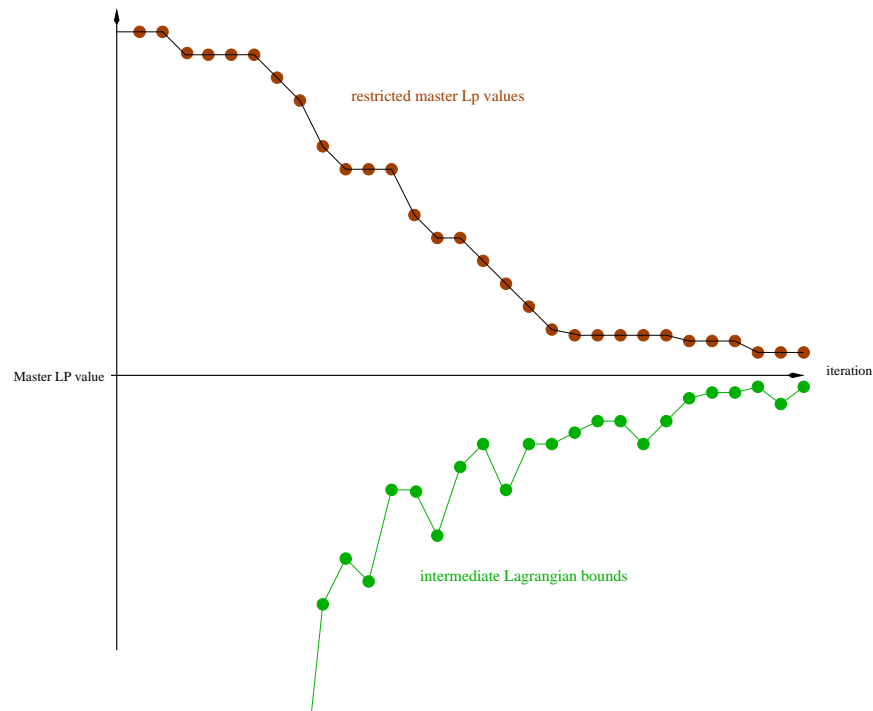
Solving the Master LP

6 Col. Gen. / Kelley: the issue of convergence



Solving the Master LP

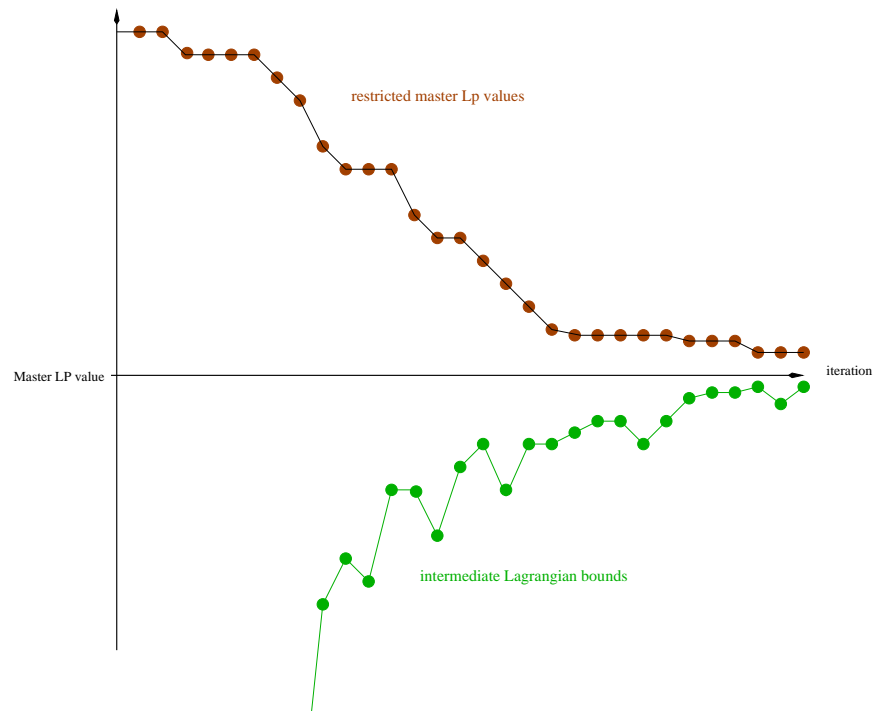
6 Col. Gen. / Kelley: the issue of convergence



△ tailing-off effect

Solving the Master LP

6 Col. Gen. / Kelley: the issue of convergence



- △ tailing-off effect
- △ heading-in effect

Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques:

Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol

Solving the Master LP

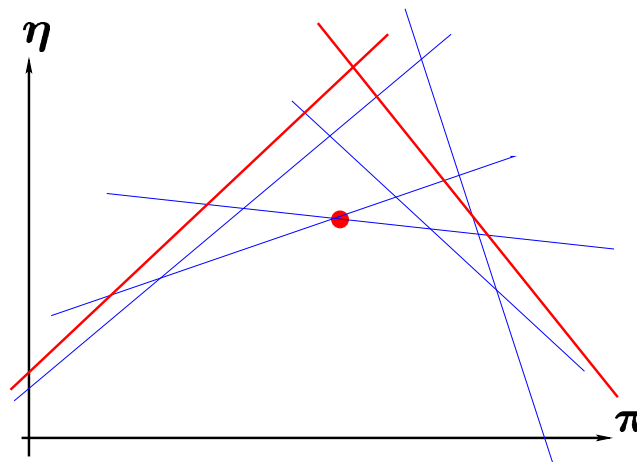
- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)

Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**

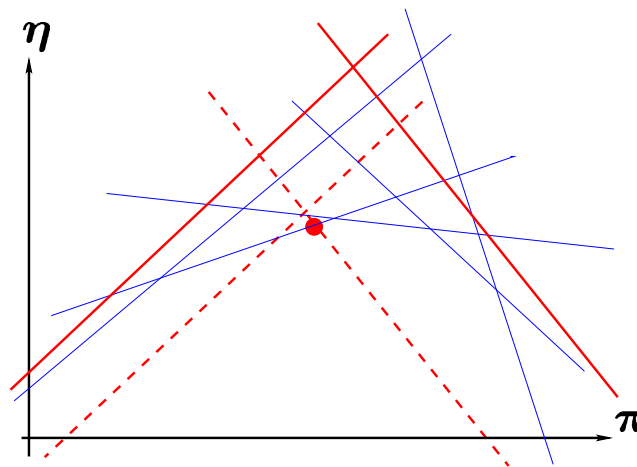
Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**



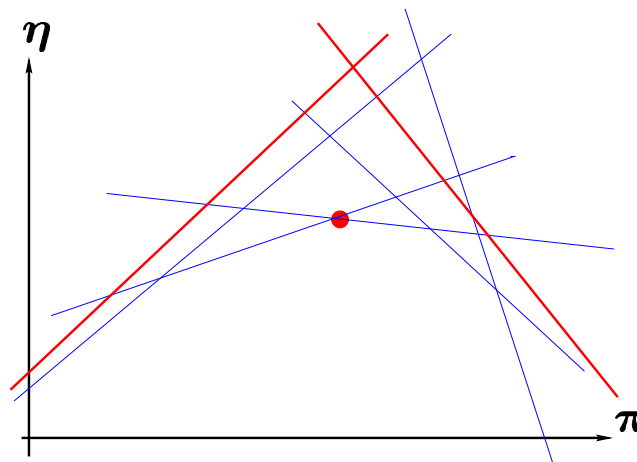
Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**



Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**



Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**
(cost initially low $\rightarrow \pi$ interior point)
 - △ use **exchange vectors** (\rightarrow dual cut)

Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**
(cost initially low $\rightarrow \pi$ interior point)
 - △ use **exchange vectors** (\rightarrow dual cut)
- ⑥ Sub-gradient algorithm

Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**
(cost initially low $\rightarrow \pi$ interior point)
 - △ use **exchange vectors** (\rightarrow dual cut)
- ⑥ Sub-gradient algorithm
- ⑥ Bundle method (Lemaréchal)

Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**
(cost initially low $\rightarrow \pi$ interior point)
 - △ use **exchange vectors** (\rightarrow dual cut)
- ⑥ Sub-gradient algorithm
- ⑥ Bundle method (Lemaréchal)
- ⑥ ACCPM (Goffin and Vial)

Solving the Master LP

- ⑥ Column Generation = Kelley
- ⑥ Stabilization techniques: **the basics**
 - △ **warm start**: initialize master f.i. with SP opt. sol
 - △ **inequality** constraint in the Master ($\rightarrow \pi \geq 0$)
 - △ include artificial **missing columns**
(cost initially low $\rightarrow \pi$ interior point)
 - △ use **exchange vectors** (\rightarrow dual cut)
- ⑥ Sub-gradient algorithm
- ⑥ Bundle method (Lemaréchal)
- ⑥ ACCPM (Goffin and Vial)

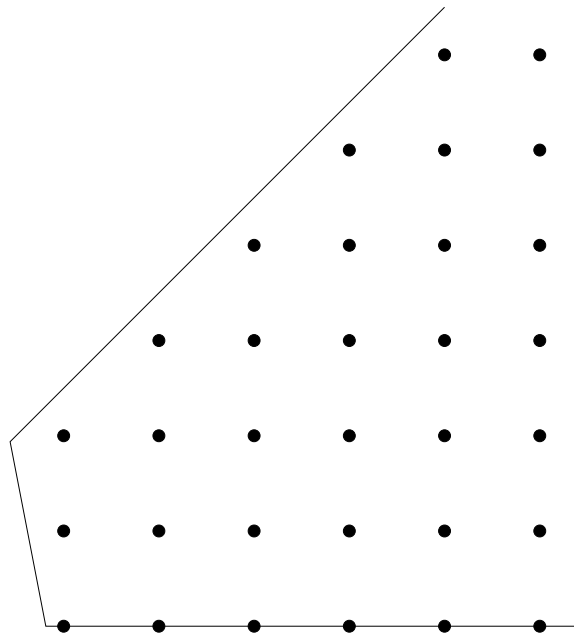
Combine with branch-and-Bound

Generating Set (1/3)



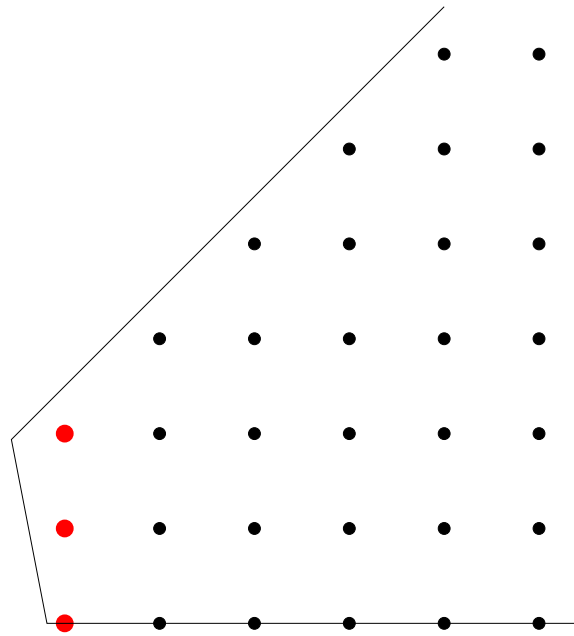
Generating Set (1/3)

1. Discretization of an unbounded IP



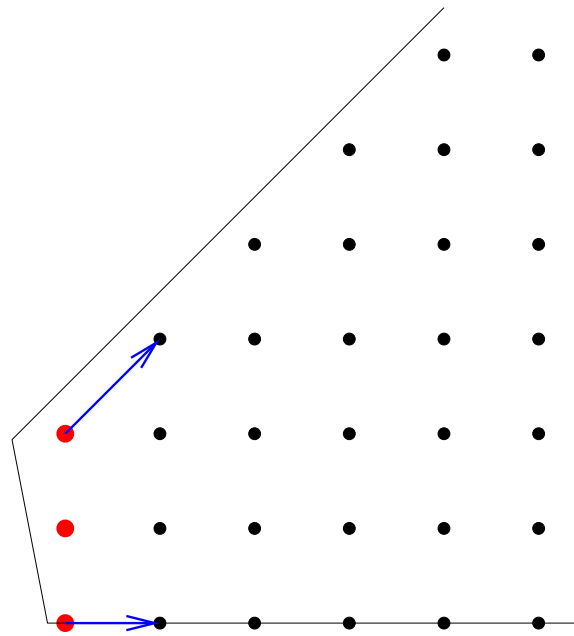
Generating Set (1/3)

1. Discretization of an unbounded IP



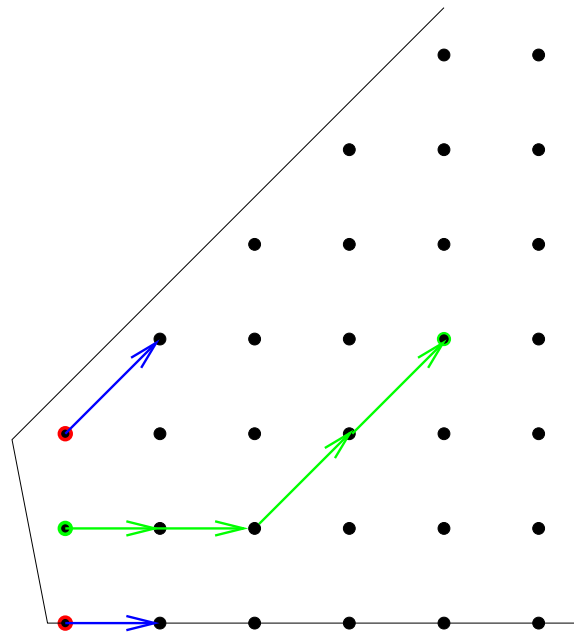
Generating Set (1/3)

1. Discretization of an unbounded IP



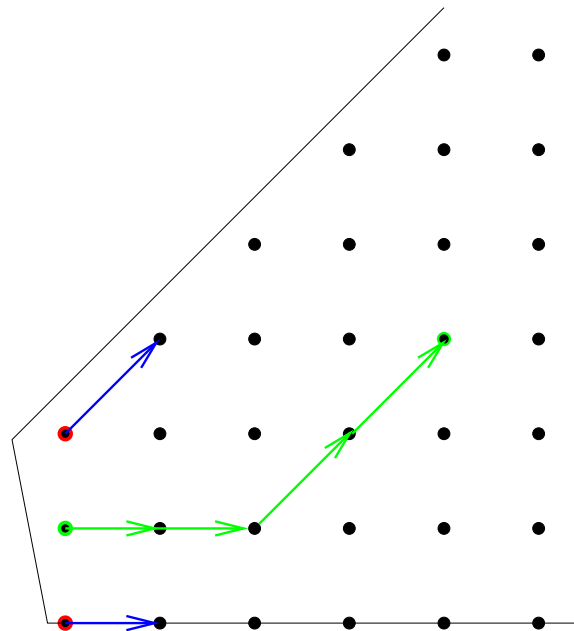
Generating Set (1/3)

1. Discretization of an unbounded IP



Generating Set (1/3)

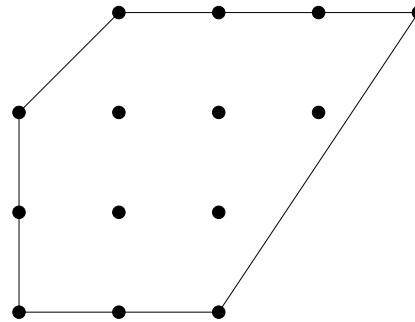
1. Discretization of an unbounded IP



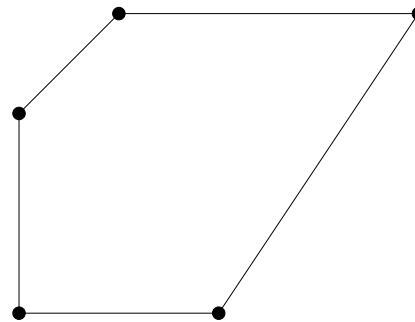
$$\left\{ y = \sum_{p \in P} p \lambda_p + \sum_{r \in R} r \lambda_r : \sum_{p \in P} \lambda_p = 1, \lambda_p \in \{0, 1\} \forall p, \lambda_r \in \mathbb{N} \forall r \right\}$$

Generating Set (2/3)

2. Discretization of an bounded IP

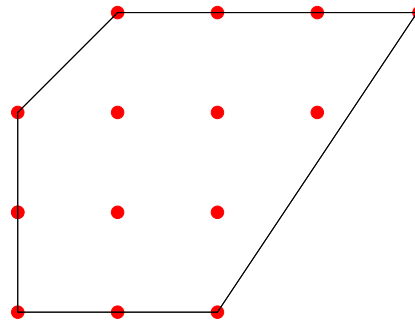


3. Convexification of a bounded IP

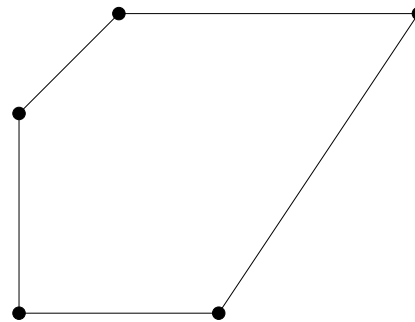


Generating Set (2/3)

4. Discretization of an bounded IP

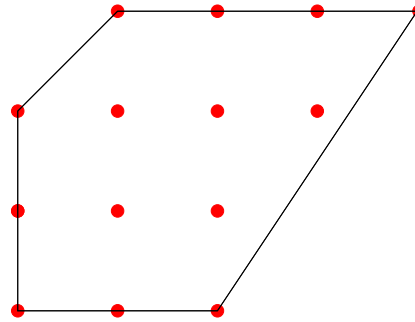


5. Convexification of a bounded IP

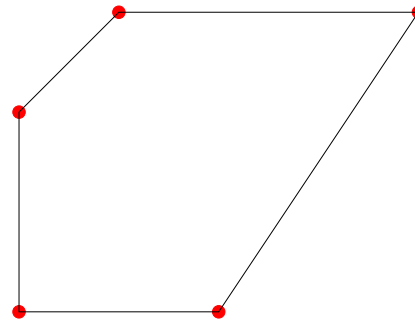


Generating Set (2/3)

6. Discretization of an bounded IP

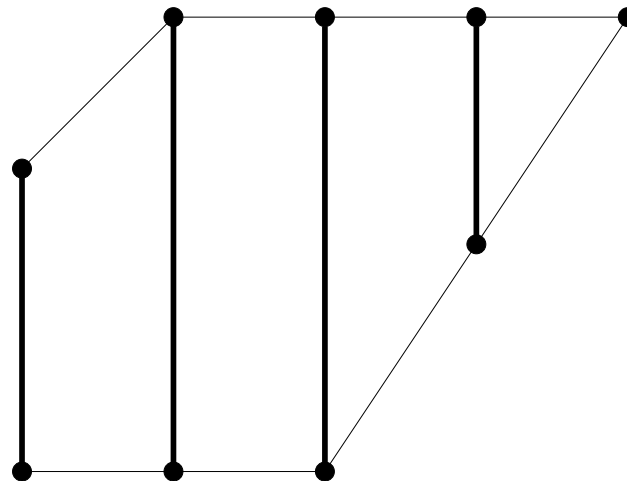


7. Convexification of a bounded IP



Generating Set (3/3)

8. Discretization of a MIP

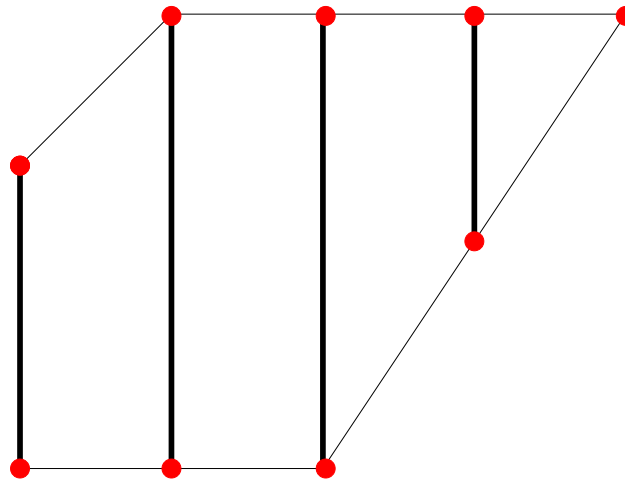


9. Its Projection in the IP variables



Generating Set (3/3)

8. Discretization of a MIP



9. Its Projection in the IP variables



Formulating Integrality restrictions

- ⑥ **Discretization approach:**
true IP re-formulation

- ⑥ **Convexification approach:**
mere LP relaxation re-formulation

Formulating Integrality restrictions

- ⑥ **Discretization approach:**
true IP re-formulation

$$\begin{array}{ll} \text{pure IP:} & \lambda_g \in \mathbf{IN} \quad \forall g \in G_d \\ \text{MIP:} & \sum_{g \in G(y)} \lambda_g \in \mathbf{IN} \quad \forall y \in G_p \end{array}$$

- ⑥ **Convexification approach:**
mere LP relaxation re-formulation

Formulating Integrality restrictions

- ⑥ **Discretization approach:**
true IP re-formulation

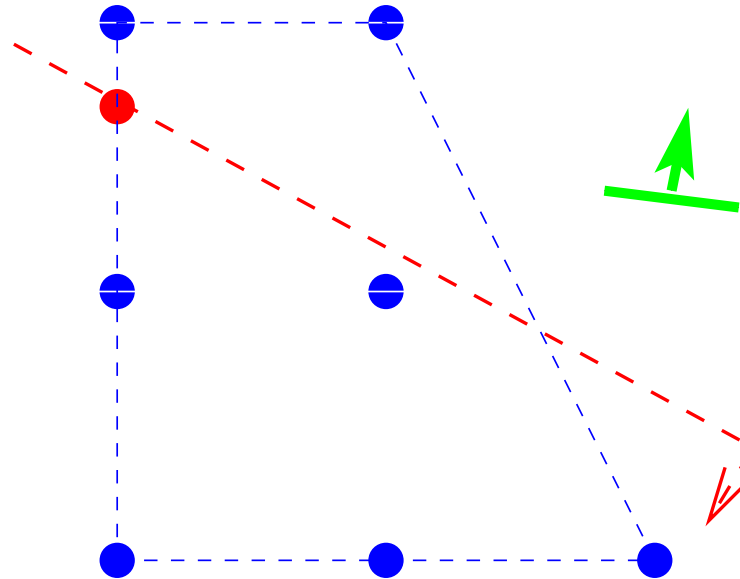
$$\begin{array}{ll} \text{pure IP:} & \lambda_g \in \mathbf{IN} \quad \forall g \in G_d \\ \text{MIP:} & \sum_{g \in G(y)} \lambda_g \in \mathbf{IN} \quad \forall y \in G_p \end{array}$$

- ⑥ **Convexification approach:**
mere LP relaxation re-formulation

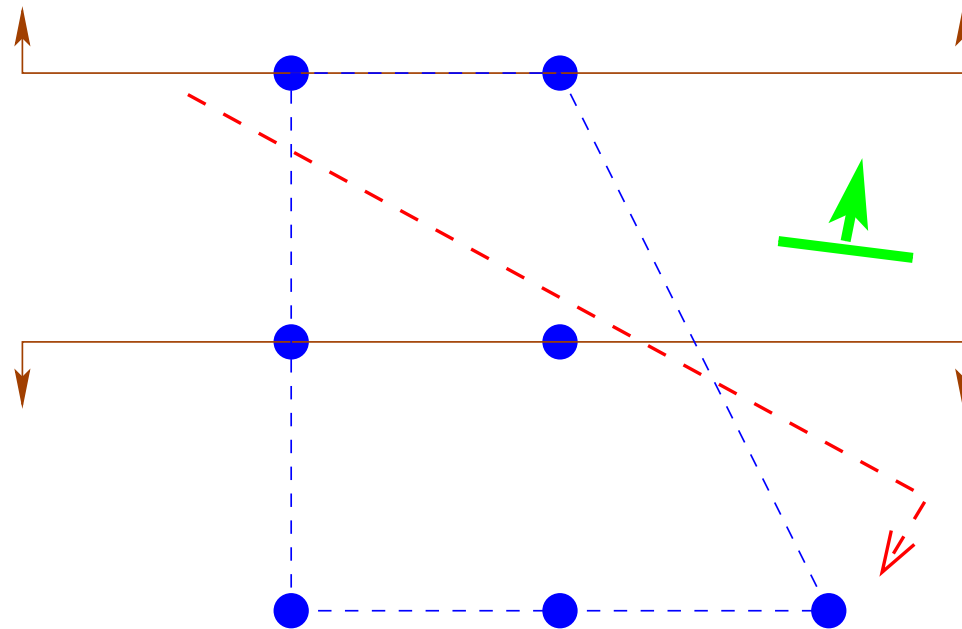
$$y = \sum_{g \in G} y^g \lambda_g \in \mathbf{IN}$$

Branching: example

Assume: pure IP, single sub-problem , $\sum_{g \in G} \lambda_g = 1$

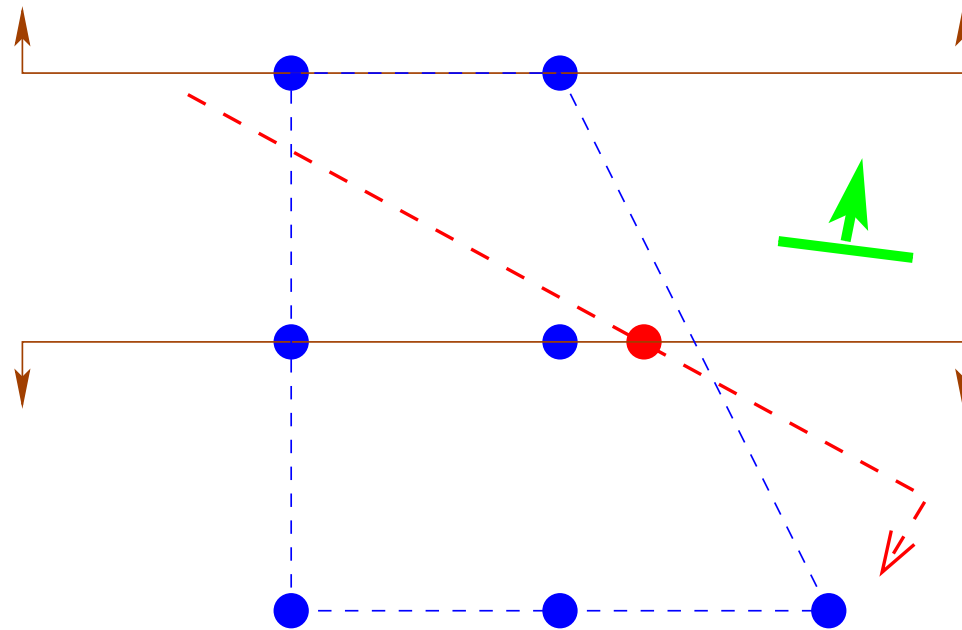


Branching: example



$$y_i \leq \lfloor \alpha_i \rfloor \quad \text{or} \quad y_i \geq \lceil \alpha_i \rceil$$

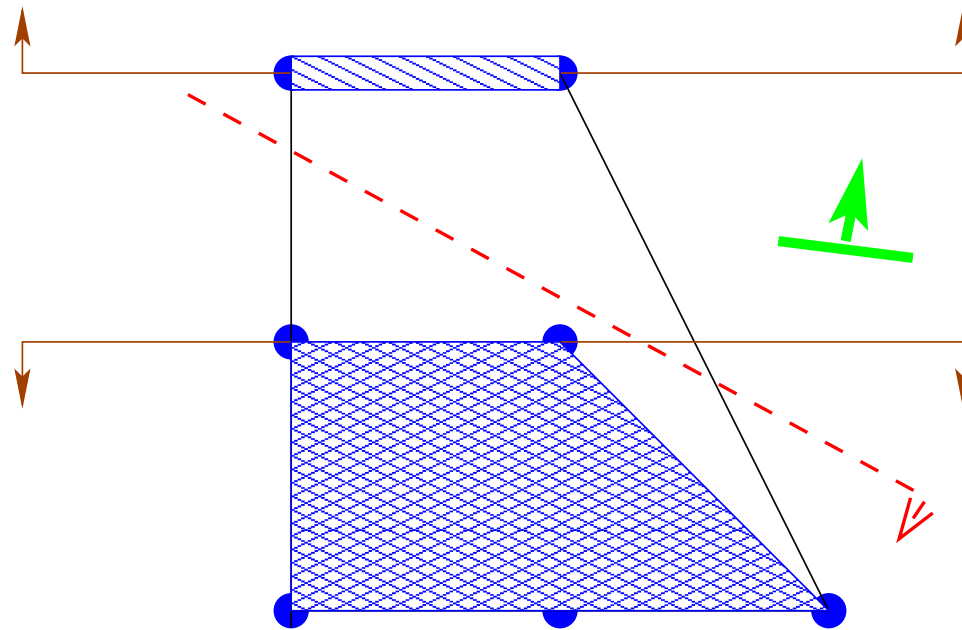
Branching: example



under convexification

$$\sum_{g \in G_c} y_i^g \lambda_g \leq \lfloor \alpha_i \rfloor \quad \text{or} \quad \sum_{g \in G_c} y_i^g \lambda_g \geq \lceil \alpha_i \rceil$$

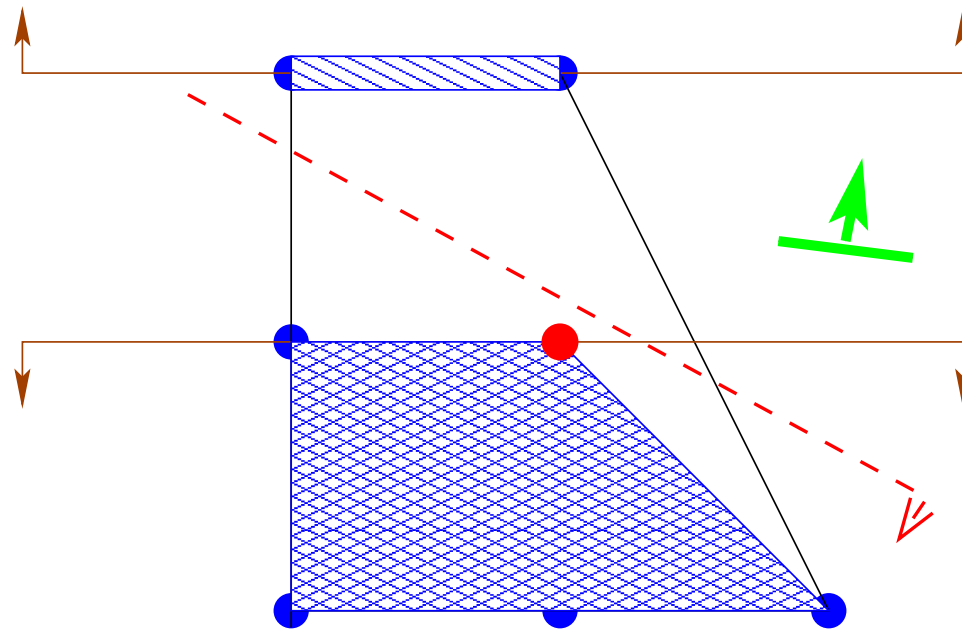
Branching: example



under discretization

$$\sum_{g \in G_d: y_i^g \geq \lceil \alpha_i \rceil} \lambda_g \leq 0 \quad \text{or} \quad \sum_{g \in G_d: y_i^g \leq \lfloor \alpha_i \rfloor} \lambda_g \leq 0$$

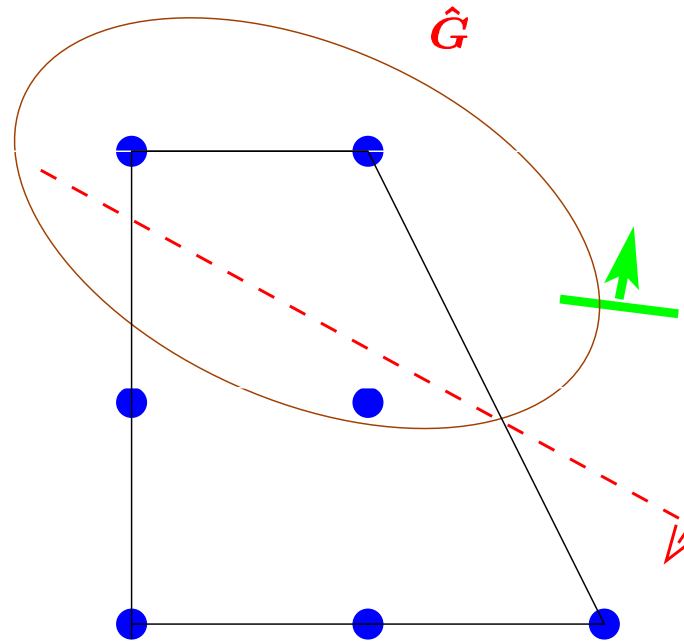
Branching: example



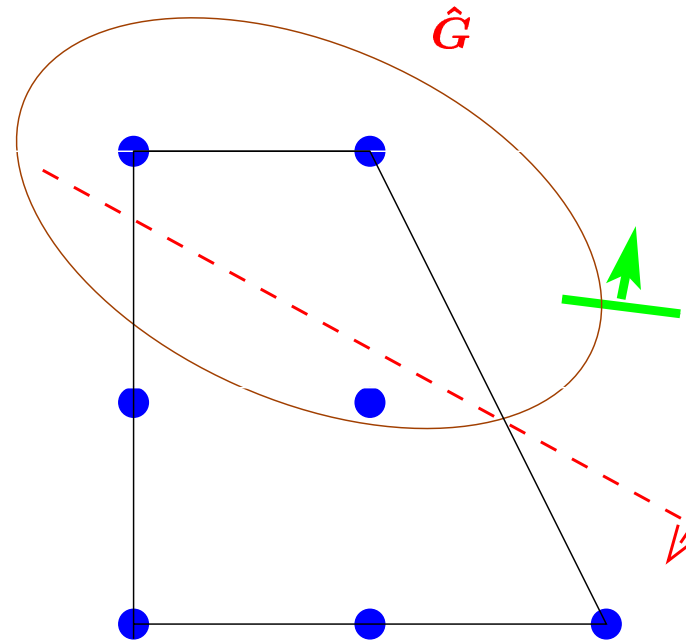
under discretization

Stronger Dual Bound

Branching: general case

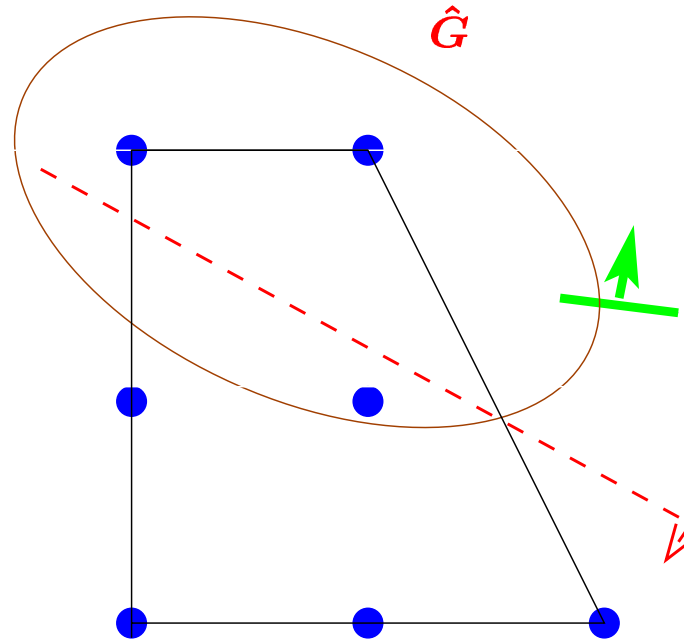


Branching: general case



$$\sum_{g \in \hat{G}} \lambda_g \leq \lfloor \alpha \rfloor \quad \text{or} \quad \sum_{g \in \hat{G}} \lambda_g \geq \lceil \alpha \rceil$$

Branching: general case



$$\sum_{g \in \hat{G}} \lambda_g \leq$$

$$\underbrace{[\alpha]}$$

$$= 0 ?$$



$$G := G \setminus \hat{G}$$

or

$$\sum_{g \in \hat{G}} \lambda_g \geq$$

$$\underbrace{[\alpha]}$$

$$= U ?$$



$$G := \hat{G}$$

Proper Columns



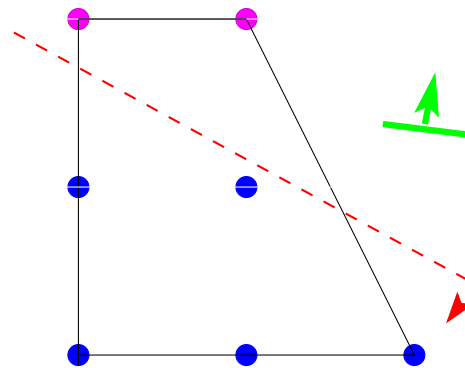
Proper Columns

$$Z^{IP} = \min \quad c y$$

$$A y \geq a$$

$$B y \geq b$$

$$y \in \mathbb{N}^p$$



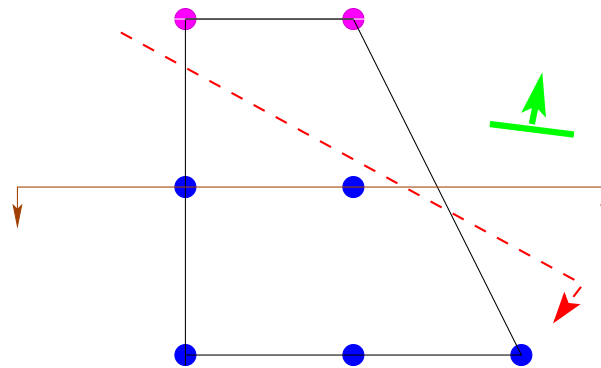
Proper Columns

$$Z^{IP} = \min \quad c y$$

$$A y \geq a$$

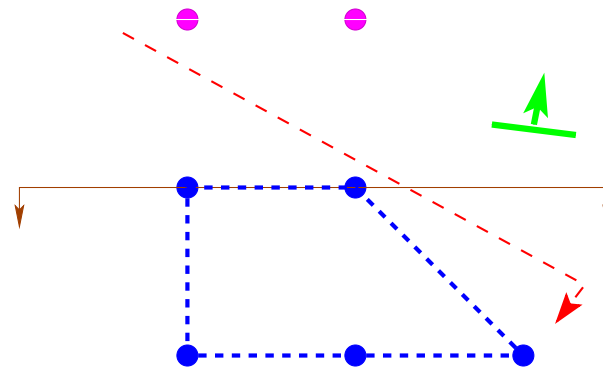
$$B y \geq b$$

$$y \in \mathbb{N}^p$$



Proper Columns

$$\begin{aligned} Z^{IP} = \min \quad & c y \\ & A y \geq a \\ & B y \geq b \\ & y \in \mathbb{N}^p \end{aligned}$$



Improved Dual Bound

Proper Columns: pseudo-definition

$$\begin{aligned} Z^{IP} = \min \quad & c y \\ & A y \geq a \\ & B y \geq b \\ & y \in \mathbb{N}^p \end{aligned}$$

$g \in G^B$ is proper if $l_A \leq g \leq u_A$
where l_A and u_A are component bounds
“implied by”
master constraints A .

Proper Columns: examples

1. Cutting Stock Problem

$$x_i \leq d_i$$

(2-d knapsack sub-problem gets strongly NP-Hard)

Proper Columns: examples

1. Cutting Stock Problem

$$x_i \leq d_i$$

(2-d knapsack sub-problem gets strongly NP-Hard)

2. Multi-Item Lot-Sizing Problem

$$x_{i,t} \leq C_t$$

(Capacitated Lot-Sizing sub-problem is NP-Hard)

Strongly Proper Columns

push pre-processing further (what if questions)

Strongly Proper Columns

push pre-processing further (what if questions)

⑥ **pseudo-definition:**

$g \in G^B$ is strongly proper if compon. bounds are s.t.

$\lambda_g = 1$ yields a *residual problem* that is *not infeasible*

Strongly Proper Columns

push pre-processing further (what if questions)

⑥ **pseudo-definition:**

$g \in G^B$ is strongly proper if compon. bounds are s.t.

$\lambda_g = 1$ yields a *residual problem* that is *not infeasible*

⑥ **Example: Multi-Item Lot-Sizing Problem**

$$x_{i,t} = C_t \Rightarrow x_{j,t} = 0 \quad \forall j \neq i$$

Strongly Proper Columns

push pre-processing further (what if questions)

⑥ pseudo-definition:

$g \in G^B$ is strongly proper if compon. bounds are s.t.

$\lambda_g = 1$ yields a *residual problem* that is *not infeasible*

⑥ Example: Multi-Item Lot-Sizing Problem

$$x_{i,t} = C_t \Rightarrow x_{j,t} = 0 \quad \forall j \neq i$$

(compute bounds that account for capacity requirement of other products)

⑥ important for rounding heuristic

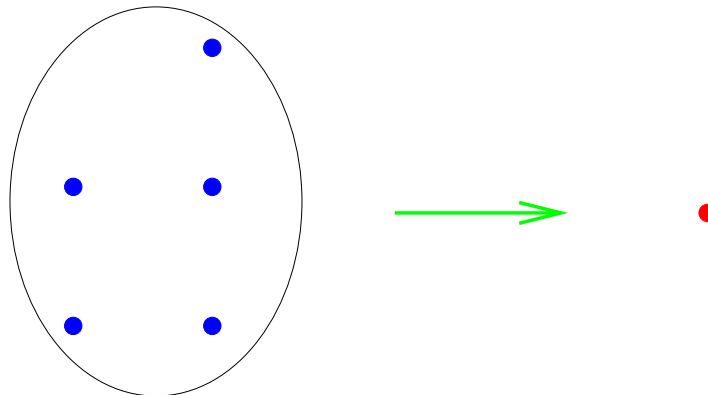
State Space relaxation

relax the definition of the generating set
easier sub-problem → weaker dual bound

State Space relaxation

relax the definition of the generating set
easier sub-problem → weaker dual bound

different
states g
of the
Universe G } mapping → base-pattern



Base-Patterns: Cutting Stock

Definition: (Fekete and Schepers, 1998)

A *mapping* $u: s \in \mathbb{R} \rightarrow u(s) \in \mathbb{R}$ is dual feasible for a CSP instance if

$$\sum_i u(s_i) g_i \leq 1 \quad \forall g \in KNP$$

where KNP is the set of feasible knapsack solution using the original sizes s_i 's.

Base-Patterns: Cutting Stock

Definition: (Fekete and Schepers, 1998)

A *mapping* $u: s \in \mathbb{R} \rightarrow u(s) \in \mathbb{R}$ is dual feasible for a CSP instance if

$$\sum_i u(s_i) g_i \leq 1 \quad \forall g \in KNP$$

where KNP is the set of feasible knapsack solution using the original sizes s_i 's.

fewer different sizes
smaller numbers } \Rightarrow easier knapsack SP

Base-Patterns: MICLS Example

Continuous Single Item Lot-Sizing sub-problem

$$\{x \in \mathbb{R}_+^T, y \in \{0, 1\}^T : \sum_{\tau=1}^t x_{\tau} \geq d_{i1t} \forall t, x_t \leq c_{it} y_t \forall t\}$$



Discrete Single Item Lot-Sizing sub-problem

$$\{y \in \{0, 1\}^T : \sum_{\tau=1}^t c_{i\tau} y_{\tau} \geq d_{i1t} \forall t\}$$

Uses of Base-Pattern Relaxation

- ⑥ relaxed $G \Rightarrow$ weaker dual bound
cheaper B-a-P node comput. **but** larger B-a-P tree

Uses of Base-Pattern Relaxation

- ⑥ relaxed $G \Rightarrow$ weaker dual bound
cheaper B-a-P node comput. **but** larger B-a-P tree
- ⑥ **warm start** for column generation:
relaxed columns acts as artificial columns

Uses of Base-Pattern Relaxation

- ⑥ relaxed $G \Rightarrow$ weaker dual bound
cheaper B-a-P node comput. **but** larger B-a-P tree
- ⑥ **warm start** for column generation:
relaxed columns acts as artificial columns
- ⑥ **Scaling approach**: iteratively refined mapping

Uses of Base-Pattern Relaxation

- ⑥ relaxed $G \Rightarrow$ weaker dual bound
cheaper B-a-P node comput. **but** larger B-a-P tree
- ⑥ **warm start** for column generation:
relaxed columns acts as artificial columns
- ⑥ **Scaling approach**: iteratively refined mapping
- ⑥ **2-stage** SP optimization: **primal-dual heuristic** for SP

Uses of Base-Pattern Relaxation

- ⑥ relaxed $G \Rightarrow$ weaker dual bound
cheaper B-a-P node comput. **but** larger B-a-P tree
- ⑥ **warm start** for column generation:
relaxed columns acts as artificial columns
- ⑥ **Scaling approach**: iteratively refined mapping
- ⑥ **2-stage** SP optimization: **primal-dual heuristic** for SP
- ⑥ reduced cost **re-optimization** from base-pattern

Uses of Base-Pattern Relaxation

- ⑥ relaxed $G \Rightarrow$ weaker dual bound
cheaper B-a-P node comput. **but** larger B-a-P tree
- ⑥ **warm start** for column generation:
relaxed columns acts as artificial columns
- ⑥ **Scaling approach**: iteratively refined mapping
- ⑥ **2-stage** SP optimization: **primal-dual heuristic** for SP
- ⑥ reduced cost **re-optimization** from base-pattern
- ⑥ include **exchange vectors** between columns sharing the same base-pattern

Analysis of the Generating Set

- ⑥ Convexification v.s. Discretization
- ⑥ Proper Columns and Strongly Proper Col.
- ⑥ State Space-Relaxation and Base-pattern
- ⑥ Dominant/Redundant Columns: **lifting**

Combining Col. Gen. with other techniques

6 Cutting planes

Combining Col. Gen. with other techniques

- ⑥ Cutting planes
- ⑥ Preprocessing, Variable fixing
(master information passed onto sub-problem)

Combining Col. Gen. with other techniques

- ⑥ Cutting planes
- ⑥ Preprocessing, Variable fixing
(master information passed onto sub-problem)
- ⑥ Primal Heuristics:
 - △ greedy
 - △ local search
 - △ rounding

Combining Col. Gen. with other techniques

- ⑥ Cutting planes
- ⑥ Preprocessing, Variable fixing
(master information passed onto sub-problem)
- ⑥ Primal Heuristics:
 - △ greedy
 - △ local search
 - △ rounding
- ⑥ Hybrid Algorithms:
f.i. **sub-gradient + col gen** (Fischetti)

a generic Branch-And-Price Code: *C++ subroutine library*

a generic Branch-And-Price Code: C++ subroutine library

⑥ implements:

1. an automatic Dantzig-Wolfe re-formulation
2. a Branch-And-Price(-and-Cut) algorithm
(+ primal heuristics)

a generic Branch-And-Price Code: C++ subroutine library

⑥ **implements:**

1. an **automatic** Dantzig-Wolfe **re-formulation**
2. a **Branch-And-Price**(-and-Cut) algorithm
(+ primal heuristics)

⑥ **User input:**

a generic Branch-And-Price Code: C++ subroutine library

⑥ **implements:**

1. an **automatic** Dantzig-Wolfe **re-formulation**
2. a **Branch-And-Price**(-and-Cut) algorithm
(+ primal heuristics)

⑥ **User input:**

1. Data definition and reading

a generic Branch-And-Price Code: C++ subroutine library

⑥ **implements:**

1. an **automatic** Dantzig-Wolfe **re-formulation**
2. a **Branch-And-Price**(-and-Cut) algorithm
(+ primal heuristics)

⑥ **User input:**

1. Data definition and reading
2. Variable and Constraint C++ Classes

a generic Branch-And-Price Code: C++ subroutine library

⑥ **implements:**

1. an **automatic** Dantzig-Wolfe **re-formulation**
2. a **Branch-And-Price**(-and-Cut) algorithm
(+ primal heuristics)

⑥ **User input:**

1. Data definition and reading
2. Variable and Constraint C++ Classes
3. Call to constructors of Variables, Constraints, Master and Sub-problems

Bapcod: MICLS Example (1/4)

```
class Period
{
    double C_t; // capacity

    Period(const Double & capacity): C_t(capacity) {}
    ~Period(){}
};

class Item
{
    double s_i; // capacity consumed in a setup
    double f_i; // setup cost
    map< Period *, Double> p_it; // production cost
    map< Period *, Double> d_it; // demand
    map< Period *, Double> c_it; // production capacity

    Item(...) {...}
    ~Item(){}
};
```

Bapcod: MICLS Example (2/4)

```
class YitGenVar: public GenericVar
{
    YitGenVar(vector<Item *> & itemPts,
              vector<Period *> & periodPts,
              map<IndexCell, SpConf *> & spConfMap)
    {
        for (vector<Item *>::iterator itemPt = ...)
            for (vector<Period *>::iterator periodPt = ...)
                {
                    new InstanciatedVar(...);
                }
    }
    ~YitGenVar(){}
};
```

Bapcod: MICLS $y_{it}^A \geq y_{it}^B$ (3/4)

```
class YABitGenConstr: public GenericConstr
{
    YABitGenConstr(vector<Item*> ..., vector<Period*> ...,
                  MasterConf* masterPtr)
    {... new InstanciadedConstr(masterPtr, ...); }

    const bool genericCoef(InstanciadedConstr * iconstrPtr,
                          InstanciadedVar * ivarPtr)
    {
        YitGenVar* YitPtr = dynamic_cast<YitGenVar*>(ivarPtr->genVarC
        if (YitPtr == NULL) return 0;
        if (iconstrPtr->id() != ivarPtr->id()) return 0;
        SpConf* SpConfPtr = dynamic_cast<SpConf *>(ivarPtr->probConf
        if (_itemSpConfPts.count(SpConfPtr)) return 1;else return -1
    }
};
```

Bapcod: MICLS Example (4/4)

```
readData();
_masterPtr = new MasterConf();
for (vector<Item *> ...) new SpConf(Item..., _masterPtr, NULL, 1);
for (vector<Period *>...) new SpConf(Period..., _masterPtr, 1, NULL);

new XitGenVar(itemPts, periodPts, itemSpConfPts);
new XitGenVar(itemPts, periodPts, periodSpConfPts);
new YitGenVar(itemPts, periodPts, itemSpConfPts);
new YitGenVar(itemPts, periodPts, periodSpConfPts);
new XABitGenConstr(itemPts, periodPts, _masterPtr);
new YABitGenConstr(itemPts, periodPts, _masterPtr);
new DemCOVitGenConstr(itemPts, periodPts, itemSpConfPts);
new XitUbGenConstr(itemPts, periodPts, itemSpConfPts);
new CAPtGenConstr(periodPts, periodSpConfPts);
new XitUbGenConstr(itemPts, periodPts, periodSpConfPts);

_masterPtr->prepareProbConfig();
_masterPtr->solve();
```

Bapcod: MICLS Preliminary Results

Instance	i6-t15	i6-t30	i12-t15	i12-t30
LP relaxation*	11019.8	14145.5	16786.0	18124.2
Cut. Plane*	37213.3	60963.2	73848.0	130177.0
DW decomp ULS	37202.0			
DW decomp CLS	37220.0	60947.0	73848.0	130180.0
DW decomp, strongly proper col	37227.0			
Lagrangian decomp	37223.0	60947.0	73854.0	130180.0
Incumbent IP sol	*37721.0	*61765.0	*74634.0	131027.0

* Belvaux and Wolsey (instances from Trigeiro et al., 1989)

Selection of sub-systems A and B

- ⑥ Tested computationally
- ⑥ Compared theoretically: *dual bound vs sub-prob. complexity* (Thizy analyses MICLS)
- ⑥ Consider duplicating constraints
- ⑥ Consider adding implicit constraints
- ⑥ Consider nested decomposition

$$\text{DW decomposition} \rightarrow \{\otimes_{t=1}^T A_{LP}^t \cap \otimes_{i=1}^I C(B^i)\}$$

$$\text{Lagrang. decomp.} \rightarrow \{\otimes_{t=1}^T C(A^t) \cap \otimes_{i=1}^I C(B^i)\}$$

$$\text{Nested decomp.} \rightarrow \{\otimes_{t=1}^T C(A^t \cap \otimes_{i=1}^I C(B^i)) \cap \otimes_{i=1}^I C(B^i)\}$$

BaPCod's *immediate future*

1.

See whether plain use of *BaPCod* + Mip solver
better than
Mip-solver applied to original formulation.

2.

Integrate “improvement techniques”
and
test their efficiency across \neq applications.

Bapcod: MICLS Preliminary Results

Instance	i6-t15	i6-t30	i12-t15	i12-t30
LP relaxation*	11019.8	14145.5	16786.0	18124.2
Cut. Plane*	37213.3	60963.2	73848.0	130177.0
DW decomp ULS	37202.0	60947.0	73848.0	130180.0
DW decomp CLS	37220.0	60947.0	73848.0	130180.0
DW decomp, strongly proper col	37227.0	60947.0	73848.0	130180.0
Lagrangian decomp	37223.0	60947.0	73854.0	130180.0
Incumbent IP sol	*37721.0	*61765.0	*74634.0	131027.0