Intermediate IP representations using value disjunctions

Matthias Köppe, Quentin Louveaux, Robert Weismantel

Otto-von-Guericke-Universität Magdeburg
Fakultät für Mathematik
Institut für Mathematische Optimierung

May 2006

Integer Programming: State of the art

$$\max c^\top x : x \in P \cap Z^n$$

Dual methods

- based on outer description of $\text{conv}(P \cap Z^n)$
- well explored
- branch-and-cut-algorithms

Primal methods

- inner descriptions of $P \cap Z^n$
- Integral Basis Method
- reformulations with new vars

$$\max c^\top x : x \in P \cap Z^n$$

Dual methods

- based on outer description of $\text{conv}(P \cap Z^n)$
- well explored
- branch-and-cut-algorithms

Primal-dual methods

- based on intermediate representations
- new variables *and* inequalities
- not explored at all

Primal methods

- inner descriptions of $P \cap Z^n$
- Integral Basis Method
- reformulations with new vars

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables
- . . . such that in the extended space dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations... but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables...
- ...such that, in the extended space, dual techniques become more powerful

Primal-dual methods vs. other methods

Column generation techniques

- extended reformulation with (exponentially many) variables
- no automatic method for general problems

Reformulations like Sherali–Adams etc.

- strong reformulations with beautiful properties
- polynomial-size reformulations. . . but huge!
- only useful when projected back into original space

Goals of the primal-dual techniques

- automatic method for general IP
- create moderately many new variables. . .
- . . . such that, in the extended space, dual techniques become more powerful

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
3. When integer infeasibility or optimality proved, fathom the node

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
3. When integer infeasibility or optimality proved, fathom the node

Integration of the Integral Basis Method into B&C

Joint work with A. Fügenschuh, A. Martin, R. Weismantel

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
    - Try to create extended formulation with improved dual bound
    - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
3. When integer infeasibility or optimality proved, fathom the node

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
3. When integer infeasibility or optimality proved, fathom the node

Integration of the Integral Basis Method into B&C

Joint work with A. Fügenschuh, A. Martin, R. Weismantel

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
3. When integer infeasibility or optimality proved, fathom the node

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
   - Compute tightened variable bounds
   - Separate a class of simple combinatorial inequalities
   - Store improved dual bound
3. When integer infeasibility or optimality proved, fathom the node

Integration of the Integral Basis Method into B&C

Joint work with A. Fügenschuh, A. Martin, R. Weismantel

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
   - Compute tightened variable bounds
   - Separate a class of simple combinatorial inequalities
   - Store improved dual bound
3. When integer infeasibility or optimality proved, fathom the node

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
   - Compute tightened variable bounds
   - Separate a class of simple combinatorial inequalities
   - Store improved dual bound
3. When integer infeasibility or optimality proved, fathom the node

Integration of the Integral Basis Method into B&C

Joint work with A. Fügenschuh, A. Martin, R. Weismantel

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
   - Compute tightened variable bounds
   - Separate a class of simple combinatorial inequalities
   - Store improved dual bound
3. When integer infeasibility or optimality proved, fathom the node

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
   - Compute tightened variable bounds
   - Separate a class of simple combinatorial inequalities
   - Store improved dual bound
3. When integer infeasibility or optimality proved, fathom the node

At nodes with no dual gain in strong branching:

1. Run Integral Basis Method with a time limit or iterations limit
   - Try to create extended formulation with improved dual bound
   - Search for improving vectors
2. Project extended formulation into original or intermediate variable space
   - Compute tightened variable bounds
   - Separate a class of simple combinatorial inequalities
   - Store improved dual bound
3. When integer infeasibility or optimality proved, fathom the node

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Integration of the Integral Basis Method into B&C

Joint work with A. Fügenschuh, A. Martin, R. Weismantel

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Current state of the project

Implemented within the branch&cut system SIP

- a sophisticated academic solver (by Alexander Martin et al.)
- available (to us) in source code form

and GYWOPT, our implementation of primal reformulation techniques

- used in computations with the Integral Basis Method (Haus, K., Weismantel, 2001/2003)
- freely available in source code form

Future work

Reimplementation in a more powerful branch&cut system

- SCIP (Tobias Achterberg)
- or one of the COIN-OR branch&cut systems

Outline of this talk

1. A very simple example. The simplification effect of reformulation.
2. The value disjunction technique. Definitions and examples.
3. The structure theorem of value disjunction.
4. The simplification effect of branching.
5. Branching on binary variables vs. branching on values.
6. Experiments with some hard problems.

Outline of this talk

1. A very simple example. The simplification effect of reformulation.
2. The value disjunction technique. Definitions and examples.
3. The structure theorem of value disjunction.
4. The simplification effect of branching.
5. Branching on binary variables vs. branching on values.
6. Experiments with some hard problems.

Outline of this talk

1. A very simple example. The simplification effect of reformulation.
2. The value disjunction technique. Definitions and examples.
3. The structure theorem of value disjunction.
4. The simplification effect of branching.
5. Branching on binary variables vs. branching on values.
6. Experiments with some hard problems.

Outline of this talk

1. A very simple example. The simplification effect of reformulation.
2. The value disjunction technique. Definitions and examples.
3. The structure theorem of value disjunction.
4. The simplification effect of branching.
5. Branching on binary variables vs. branching on values.
6. Experiments with some hard problems.

Outline of this talk

1. A very simple example. The simplification effect of reformulation.
2. The value disjunction technique. Definitions and examples.
3. The structure theorem of value disjunction.
4. The simplification effect of branching.
5. Branching on binary variables vs. branching on values.
6. Experiments with some hard problems.

Outline of this talk

1. A very simple example. The simplification effect of reformulation.
2. The value disjunction technique. Definitions and examples.
3. The structure theorem of value disjunction.
4. The simplification effect of branching.
5. Branching on binary variables vs. branching on values.
6. Experiments with some hard problems.

Intermediate representation of multi knapsack problems

An example

Consider the set $x \in \{0,1\}^8$ such that

$$8x_0 - x_1 - 2x_2 - 3x_3 - 4x_4 - 5x_5 - 6x_6 - 7x_7 \leq 0.$$

Convex hull: 13 non-trivial facets

$$\begin{aligned}
x_0 \qquad\qquad - x_3 \qquad - x_5 - x_6 - x_7 &\leq 0\\
x_0 \qquad\qquad\qquad - x_4 - x_5 - x_6 - x_7 &\leq 0\\
x_0 - x_1 - x_2 \qquad\qquad - x_5 - x_6 - x_7 &\leq 0\\
x_0 - x_1 \qquad - x_3 - x_4 \qquad - x_6 - x_7 &\leq 0\\
x_0 \qquad - x_2 - x_3 - x_4 - x_5 \qquad - x_7 &\leq 0\\
x_0 \qquad - x_2 - x_3 - x_4 \qquad - x_6 - x_7 &\leq 0\\
x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 \qquad &\leq 0\\
2x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 &\leq 0\\
2x_0 \qquad - x_2 - x_3 - x_4 - x_5 - x_6 - 2x_7 &\leq 0\\
2x_0 - x_1 \qquad - x_3 - x_4 - x_5 - 2x_6 - 2x_7 &\leq 0\\
3x_0 - x_1 - x_2 - x_3 - x_4 - 2x_5 - 2x_6 - 2x_7 &\leq 0\\
3x_0 - x_1 - x_2 - 2x_3 - 2x_4 - x_5 - 2x_6 - 2x_7 &\leq 0\\
5x_0 - x_1 - x_2 - 2x_3 - 2x_4 - 3x_5 - 4x_6 - 4x_7 &\leq 0
\end{aligned}$$

Consider the set $x \in \{0,1\}^8$ such that

$$8x_0 - x_1 - 2x_2 - 3x_3 - 4x_4 - 5x_5 - 6x_6 - 7x_7 \leq 0.$$

Convex hull: 13 non-trivial facets

$$\begin{array}{lllllllll}
x_0 & & & -x_3 & & -x_5 & -x_6 & -x_7 & \leq 0 \\
x_0 & & & & -x_4 & -x_5 & -x_6 & -x_7 & \leq 0 \\
x_0 & -x_1 & -x_2 & & & -x_5 & -x_6 & -x_7 & \leq 0 \\
x_0 & -x_1 & & -x_3 & -x_4 & & -x_6 & -x_7 & \leq 0 \\
x_0 & & -x_2 & -x_3 & -x_4 & -x_5 & & -x_7 & \leq 0 \\
x_0 & & -x_2 & -x_3 & -x_4 & & -x_6 & -x_7 & \leq 0 \\
x_0 & -x_1 & -x_2 & -x_3 & -x_4 & -x_5 & -x_6 & & \leq 0 \\
2x_0 & -x_1 & -x_2 & -x_3 & -x_4 & -x_5 & -x_6 & -x_7 & \leq 0 \\
2x_0 & & -x_2 & -x_3 & -x_4 & -x_5 & -x_6 & -2x_7 & \leq 0 \\
2x_0 & -x_1 & & -x_3 & -x_4 & -x_5 & -2x_6 & -2x_7 & \leq 0 \\
3x_0 & -x_1 & -x_2 & -x_3 & -x_4 & -2x_5 & -2x_6 & -2x_7 & \leq 0 \\
3x_0 & -x_1 & -x_2 & -2x_3 & -2x_4 & -x_5 & -2x_6 & -2x_7 & \leq 0 \\
5x_0 & -x_1 & -x_2 & -2x_3 & -2x_4 & -3x_5 & -4x_6 & -4x_7 & \leq 0
\end{array}$$

An intermediate representation:

Introduce new variables for the subsets $\{1,2\}$ and $\{3,4\}$.

Reformulation

$$\begin{aligned}
8x_0 - x_1 - 2x_2 - 3x_3 - 4x_4 - 5x_5 - 6x_6 - 7x_7 - 3x_9 - 7x_{10} &\le 0 \\
x_1 + x_2 + x_9 &\le 1 \\
x_3 + x_4 + x_{10} &\le 1
\end{aligned}$$

Convex hull: 9 non-trivial facets

$$\begin{aligned}
x_0 - x_5 - x_6 - x_7 - x_{10} &\le 0 \\
x_0 - x_1 - x_2 - x_5 - x_6 - x_7 - x_9 &\le 0 \\
x_0 - x_3 - x_4 - x_6 - x_7 - x_9 - x_{10} &\le 0 \\
x_0 - x_2 - x_3 - x_4 - x_5 - x_7 - x_9 - x_{10} &\le 0 \\
x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_9 - x_{10} &\le 0 \\
2x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 - x_9 - x_{10} &\le 0 \\
2x_0 - x_2 - x_3 - x_4 - x_5 - x_6 - 2x_7 - x_9 - 2x_{10} &\le 0 \\
x_3 + x_4 + x_{10} &\le 1 \\
x_1 + x_2 + x_9 &\le 1
\end{aligned}$$

An intermediate representation:

Introduce new variables for the subsets $\{1,2\}$ and $\{3,4\}$.

Reformulation

$$\begin{aligned}
8x_0 - x_1 - 2x_2 - 3x_3 - 4x_4 - 5x_5 - 6x_6 - 7x_7 - 3x_9 - 7x_{10} &\le 0\\
x_1 + x_2 + x_9 &\le 1\\
x_3 + x_4 + x_{10} &\le 1
\end{aligned}$$

Convex hull: 9 non-trivial facets

$$\begin{aligned}
x_0 - x_5 - x_6 - x_7 - x_{10} &\le 0\\
x_0 - x_1 - x_2 - x_5 - x_6 - x_7 - x_9 &\le 0\\
x_0 - x_3 - x_4 - x_6 - x_7 - x_9 - x_{10} &\le 0\\
x_0 - x_2 - x_3 - x_4 - x_5 - x_7 - x_9 - x_{10} &\le 0\\
x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_9 - x_{10} &\le 0\\
\mathbf{2}x_0 - x_1 - x_2 - x_3 - x_4 - x_5 - x_6 - x_7 - x_9 - x_{10} &\le 0\\
\mathbf{2}x_0 - x_2 - x_3 - x_4 - x_5 - x_6 - \mathbf{2}x_7 - x_9 - \mathbf{2}x_{10} &\le 0\\
+ x_3 + x_4 + x_{10} &\le 1\\
x_1 + x_2 + x_9 &\le 1
\end{aligned}$$

How to obtain intermediate representations?

An example

$$3x_1 + 3x_2 + 3x_3 + 4x_4 + 5x_5 \leq 9. \qquad x_i \in 0, 1$$

Two blocks and six new variables

Block N_1	$\{1, 2, 3\}$	Values: 3,6,9	New variables: y_3, y_6, y_9
Block N_2	$\{4, 5\}$	Values: 4,5,9	New variables: z_4, z_5, z_9

Reformulation

$$3y_3 + 6y_6 + 9y_9 + 4z_4 + 5z_5 + 9z_9 \leq 9$$
$$3x_1 + 3x_2 + 3x_3 = 3y_3 + 6y_6 + 9y_9$$
$$4x_4 + 5x_5 = 4z_4 + 5z_5 + 9z_9$$
$$y_3 + y_6 + y_9 \leq 1$$
$$z_4 + z_5 + z_9 \leq 1$$

How to obtain intermediate representations?

An example

$$3x_1 + 3x_2 + 3x_3 + 4x_4 + 5x_5 \leq 9. \qquad x_i \in 0, 1$$

Two blocks and six new variables

Block N_1	$\{1, 2, 3\}$	Values: 3,6,9	New variables: y_3, y_6, y_9
Block N_2	$\{4, 5\}$	Values: 4,5,9	New variables: z_4, z_5, z_9

Reformulation

$$3y_3 + 6y_6 + 9y_9 + 4z_4 + 5z_5 + 9z_9 \leq 9$$
$$3x_1 + 3x_2 + 3x_3 = 3y_3 + 6y_6 + 9y_9$$
$$4x_4 + 5x_5 = 4z_4 + 5z_5 + 9z_9$$
$$y_3 + y_6 + y_9 \leq 1$$
$$z_4 + z_5 + z_9 \leq 1$$

How to obtain intermediate representations?

An example

$$3x_1 + 3x_2 + 3x_3 + 4x_4 + 5x_5 \leq 9. \qquad x_i \in 0, 1$$

Two blocks and six new variables

Block N_1 $\{1, 2, 3\}$ Values: 3,6,9 New variables: y_3, y_6, y_9

Block N_2 $\{4, 5\}$ Values: 4,5,9 New variables: z_4, z_5, z_9

Reformulation

$$3y_3 + 6y_6 + 9y_9 + 4z_4 + 5z_5 + 9z_9 \leq 9$$
$$3x_1 + 3x_2 + 3x_3 = 3y_3 + 6y_6 + 9y_9$$
$$4x_4 + 5x_5 = 4z_4 + 5z_5 + 9z_9$$
$$y_3 + y_6 + y_9 \leq 1$$
$$z_4 + z_5 + z_9 \leq 1$$

How to obtain intermediate representations?

starting point: a knapsack relaxation, for instance

$$P = \operatorname{conv}\{x \in \{0,1\}^n : \sum_{j=1}^{n} a_j x_j \le b\}$$

one tool: value disjunctions

Partition $N = \{1, \dots, n\}$ into subsets $N_1, \dots, N_K$.

Reformulation based on N_i

Let $\{d_1, \dots, d_{n_i}\} = \{\sum_{i \in S} a_i \mid S \subseteq N_i\}$. For each value d_k we introduce a binary variable $y^{N_i,k}$.

linking constraints:

$$\sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} d_k y^{N_i,k}$$

packing constraints:

$$\sum_{k=1}^{n_i} y^{N_i,k} \le 1$$

How to obtain intermediate representations?

starting point: a knapsack relaxation, for instance

$$P = \operatorname{conv}\{x \in \{0,1\}^n : \sum_{j=1}^{n} a_j x_j \leq b\}$$

one tool: value disjunctions

Partition $N = \{1, \ldots, n\}$ into subsets $N_1, \ldots, N_K$.

Reformulation based on N_i

Let $\{d_1, \ldots, d_{n_i}\} = \{\sum_{i \in S} a_i \mid S \subseteq N_i\}$. For each value d_k we introduce a binary variable $y^{N_i,k}$.

linking constraints:

$$\sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} d_k y^{N_i,k}$$

packing constraints:

$$\sum_{k=1}^{n_i} y^{N_i,k} \leq 1$$

How to obtain intermediate representations?

starting point: a knapsack relaxation, for instance

$$P = \operatorname{conv}\{x \in \{0,1\}^n : \sum_{j=1}^{n} a_j x_j \leq b\}$$

one tool: value disjunctions

Partition $N = \{1, \ldots, n\}$ into subsets $N_1, \ldots, N_K$.

Reformulation based on N_i

Let $\{d_1, \ldots, d_{n_i}\} = \{\sum_{i \in S} a_i \mid S \subseteq N_i\}$. For each value d_k we introduce a binary variable $y^{N_i,k}$.

linking constraints:

$$\sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} d_k y^{N_i,k}$$

packing constraints:

$$\sum_{k=1}^{n_i} y^{N_i,k} \leq 1$$

Why choose value disjunction?

An example

$$3x_1 + 3x_2 + 3x_3 + 3x_4 + 4x_5 + 7x_6 + 8x_7 + 9x_8 + 13x_9 + 15x_{10} \leq 45.$$

Formulation	Equations	# Facets
original		328
integer expansion	$x_1 + x_2 + x_3 + x_4 = z$	328
binary expansion	$x_1 + x_2 + x_3 + x_4 = z_1 + 2z_2 + 4z_3$	217
value disjunction	$x_1 + x_2 + x_3 + x_4 = z_1 + 2z_2 + 3z_3 + 4z_4$ $z_1 + z_2 + z_3 + z_4 \leq 1$	77

Structural theorem for value disjunctions

An example with its extended formulation

$$X = \{x \in \{0,1,2\}^4 : x_1 + x_2 + 2x_3 + 3x_4 \leq 7\}.$$

$$X = \mathrm{Proj}_x\{(x,y) \in \{0,1,2\}^4 \times \{0,1\}^4 : \begin{array}{l} y_1 + 2y_2 + 3y_3 + 4y_4 + 2x_3 + 3x_4 \leq 7 \\ x_1 + x_2 = y_1 + 2y_2 + 3y_3 + 4y_4 \\ y_1 + y_2 + y_3 + y_4 \leq 1\}. \end{array}$$

The convex hull is the "intersection" of two polyhedra

The linking polyhedron

$$V_1 = \{(x_1,x_2,y) | \begin{array}{l} x_1 + x_2 = y_1 + 2y_2 + 3y_3 + 4y_4 \\ y_1 + y_2 + y_3 + y_4 \leq 1 \quad \} \end{array}$$

The aggregated polyhedron

$$Q = \{(x_3,x_4,y) | \begin{array}{l} y_1 + 2y_2 + 3y_3 + 4y_4 + 2x_3 + 3x_4 \leq 7 \\ y_1 + y_2 + y_3 + y_4 \leq 1 \quad \} \end{array}$$

Structural theorem for value disjunctions

An example with its extended formulation

$$X = \{x \in \{0,1,2\}^4 : x_1 + x_2 + 2x_3 + 3x_4 \leq 7\}.$$

$$X = \text{Proj}_x \{(x,y) \in \{0,1,2\}^4 \times \{0,1\}^4 : \begin{aligned} &y_1 + 2y_2 + 3y_3 + 4y_4 + 2x_3 + 3x_4 \leq 7 \\ &x_1 + x_2 = y_1 + 2y_2 + 3y_3 + 4y_4 \\ &y_1 + y_2 + y_3 + y_4 \leq 1\}. \end{aligned}$$

The convex hull is the "intersection" of two polyhedra

The linking polyhedron

$$V_1 = \{(x_1, x_2, y) | \begin{aligned} &x_1 + x_2 = y_1 + 2y_2 + 3y_3 + 4y_4 \\ &y_1 + y_2 + y_3 + y_4 \leq 1 \quad \} \end{aligned}$$

The aggregated polyhedron

$$Q = \{(x_3, x_4, y) | \begin{aligned} &y_1 + 2y_2 + 3y_3 + 4y_4 + 2x_3 + 3x_4 \leq 7 \\ &y_1 + y_2 + y_3 + y_4 \leq 1 \quad \} \end{aligned}$$

Structural theorem for value disjunction

The convex hull of the extended formulation

Nr.	c_1	c_2	c_3	c_4	d_1	d_2	d_3	d_4	c_0	Origin
(1)	0	-1	0	0	0	0	1	2	0	V_1
(2)	0	-1	0	0	1	2	2	2	0	V_1
(3)	0	0	0	0	1	1	1	1	1	Q/V_1
(4)	0	0	1	0	0	0	0	1	2	Q
(5)	0	0	0	1	0	1	1	1	2	Q
(6)	0	0	1	1	1	1	1	2	3	Q
(7)	0	0	1	2	0	1	2	2	4	Q
(8)	1	1	0	0	-1	-2	-3	-4	0	V_1

Structural theorem for value disjunctions

value disjunction polytope

$$V_i = \operatorname{conv}\Big\{ (x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \Big\}.$$

aggregated polytope

$$Q = \operatorname{conv}\Big\{ y \in \{0,1\}^{n_1+\cdots+n_K} : \sum_{i=1}^{K} \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k} \le b, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \quad \forall i \Big\}$$

Theorem (structural theorem)

$$P = \Big\{ x \in [0,1]^n : \textit{there is } y \in [0,1]^{n_1+\cdots+n_K} \textit{ such that } (x^{N_i}, y^{N_i}) \in V_i \textit{ for } i = 1,\ldots,K \textit{ and } y \in Q \Big\}.$$

Structural theorem for value disjunctions

value disjunction polytope

$$V_i = \operatorname{conv}\Big\{ (x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \Big\}.$$

aggregated polytope

$$Q = \operatorname{conv}\Big\{ y \in \{0,1\}^{n_1+\cdots+n_K} : \sum_{i=1}^{K} \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k} \le b, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \quad \forall i \Big\}$$

Theorem (structural theorem)

$$P = \Big\{ x \in [0,1]^n : \textit{there is } y \in [0,1]^{n_1+\cdots+n_K} \textit{ such that } (x^{N_i}, y^{N_i}) \in V_i \textit{ for } i = 1,\ldots,K \textit{ and } y \in Q \Big\}.$$

Structural theorem for value disjunctions

value disjunction polytope

$$V_i = \operatorname{conv}\left\{ (x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \right\}.$$

aggregated polytope

$$Q = \operatorname{conv}\left\{ y \in \{0,1\}^{n_1+\cdots+n_K} : \sum_{i=1}^{K} \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k} \le b, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \quad \forall i \right\}$$

Theorem (structural theorem)

$$P = \left\{ x \in [0,1]^n : \text{there is } y \in [0,1]^{n_1+\cdots+n_K} \text{ such that } (x^{N_i}, y^{N_i}) \in V_i \text{ for } i = 1,\dots,K \text{ and } y \in Q \right\}.$$

Structural theorem for value disjunctions

value disjunction polytope

$$V_i = \operatorname{conv}\Big\{ (x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \Big\}.$$

aggregated polytope

$$Q = \operatorname{conv}\Big\{ y \in \{0,1\}^{n_1+\dots+n_K} : \sum_{i=1}^{K} \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k} \le b, \quad \sum_{k=1}^{n_i} y^{N_i,k} \le 1 \quad \forall i \Big\}$$

Theorem (structural theorem)

$$P = \Big\{ x \in [0,1]^n : \textit{there is } y \in [0,1]^{n_1+\dots+n_K} \textit{ such that } (x^{N_i}, y^{N_i}) \in V_i \textit{ for } i = 1,\dots,K \textit{ and } y \in Q \Big\}.$$

Structural theorem for value disjunctions

value disjunction polytope

$$V_i = \operatorname{conv}\Big\{ (x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \leq 1 \Big\}.$$

aggregated polytope

$$Q = \operatorname{conv}\Big\{ y \in \{0,1\}^{n_1+\cdots+n_K} : \sum_{i=1}^{K} \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k} \leq b, \quad \sum_{k=1}^{n_i} y^{N_i,k} \leq 1 \quad \forall i \Big\}$$

Theorem (structural theorem)

$$P = \Big\{ x \in [0,1]^n : \text{there is } y \in [0,1]^{n_1+\cdots+n_K} \text{ such that } (x^{N_i}, y^{N_i}) \in V_i \text{ for } i = 1,\ldots,K \text{ and } y \in Q \Big\}.$$

value disjunction polytope

$$V_i = \operatorname{conv}\Big\{ (x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} a_j x_j = \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \leq 1 \Big\}.$$

aggregated polytope

$$Q = \operatorname{conv}\Big\{ y \in \{0,1\}^{n_1 + \cdots + n_K} : \sum_{i=1}^{K} \sum_{k=1}^{n_i} a(y^{N_i,k}) y^{N_i,k} \leq b, \quad \sum_{k=1}^{n_i} y^{N_i,k} \leq 1 \quad \forall i \Big\}$$

Theorem (structural theorem)

$$P = \Big\{ x \in [0,1]^n : \textit{there is } y \in [0,1]^{n_1+\cdots+n_K} \textit{ such that } (x^{N_i}, y^{N_i}) \in V_i \textit{ for } i = 1, \ldots, K \textit{ and } y \in Q \Big\}.$$

The value disjunction polytope V_i: The cardinality case

Theorem

$$V_i = \operatorname{conv}\{(x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} x_j = \sum_{k=1}^{n_i} k y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \leq 1\}.$$

is completely described by non-negativity constraints and:

$$\sum_{j \in N_i} x_j = \sum_{k=1}^{n_i} k y^{N_i,k}$$

$$\sum_{j \in T} x_j - \sum_{k=1}^{|T|} k y_k - \sum_{k=|T|+1}^{n_i} |T| y_k \leq 0 \qquad \text{for } \emptyset \neq T \subset N_i$$

$$\sum_{k=1}^{n_i} y^{N_i,k} \leq 1$$

Theorem

The separation problem over V_i can be solved in polynomial time.

The value disjunction polytope V_i: The cardinality case

Theorem

$$V_i = \text{conv}\{(x^{N_i}, y^{N_i}) \in \{0,1\}^{|N_i|} \times \{0,1\}^{n_i} : \sum_{j \in N_i} x_j = \sum_{k=1}^{n_i} k y^{N_i,k}, \quad \sum_{k=1}^{n_i} y^{N_i,k} \leq 1\}.$$

is completely described by non-negativity constraints and:

$$\sum_{j \in N_i} x_j = \sum_{k=1}^{n_i} k y^{N_i,k}$$

$$\sum_{j \in T} x_j - \sum_{k=1}^{|T|} k y_k - \sum_{k=|T|+1}^{n_i} |T| y_k \leq 0 \qquad \text{for } \emptyset \neq T \subset N_i$$

$$\sum_{k=1}^{n_i} y^{N_i,k} \leq 1$$

Theorem

The separation problem over V_i can be solved in polynomial time.

The knapsack with three distinct coefficients

The problem

$$\sum_{j\in N_1} \mu x_j + \sum_{j\in N_2} \lambda x_j + \sum_{j\in N_3} \sigma x_j \le \beta,$$

An extended formulation

$$\sum_{j\in N_1} \mu x_j + \sum_{j\in N_2} \lambda x_j + \sum_{j\in N_3} \sigma x_j \le \beta$$

$$\sum_{j\in N_i} x_j = \sum_{k=1}^{|N_i|} k y_k^i \qquad \text{for } i = 1,2,3$$

$$\sum_{k=1}^{|N_i|} y_k^i \le 1 \qquad \text{for } i = 1,2,3$$

$$x \in \{0,1\}^{|N_1|+|N_2|+|N_3|}$$

$$y^i \in \{0,1\}^{|N_i|} \qquad \text{for } i = 1,2,3.$$

The problem

$$\sum_{j\in N_1} \mu x_j + \sum_{j\in N_2} \lambda x_j + \sum_{j\in N_3} \sigma x_j \le \beta,$$

An extended formulation

$$\sum_{j\in N_1} \mu x_j + \sum_{j\in N_2} \lambda x_j + \sum_{j\in N_3} \sigma x_j \le \beta$$

$$\sum_{j\in N_i} x_j = \sum_{k=1}^{|N_i|} k y_k^i \qquad \text{for } i = 1,2,3$$

$$\sum_{k=1}^{|N_i|} y_k^i \le 1 \qquad \text{for } i = 1,2,3$$

$$x \in \{0,1\}^{|N_1|+|N_2|+|N_3|}$$

$$y^i \in \{0,1\}^{|N_i|} \qquad \text{for } i = 1,2,3.$$

The knapsack with three distinct coefficients

The aggregated polyhedron

$$\mu \sum_{k=1}^{|N_1|} k y^{N_1,k} + \lambda \sum_{k=1}^{|N_2|} k y^{N_2,k} + \sigma \sum_{k=1}^{|N_3|} k y^{N_3,k} \leq \beta$$

$$\sum_{k=1}^{|N_i|} y^{N_i,k} \leq 1 \qquad \text{for } i = 1, 2, 3$$

$$y^{N_i} \in \{0,1\}^{|N_i|} \qquad \text{for } i = 1, 2, 3.$$

- Let $\{v^1, \ldots, v^p\} \subseteq \{0,1\}^{|N_1|+|N_2|+|N_3|}$ be all the vertices of the aggregated polyhedron.
- Notice that $p \leq (1 + |N_1|) \cdot (1 + |N_2|) \cdot (1 + |N_3|)$.

The aggregated polyhedron

$$\mu \sum_{k=1}^{|N_1|} k y^{N_1,k} + \lambda \sum_{k=1}^{|N_2|} k y^{N_2,k} + \sigma \sum_{k=1}^{|N_3|} k y^{N_3,k} \leq \beta$$

$$\sum_{k=1}^{|N_i|} y^{N_i,k} \leq 1 \qquad \text{for } i = 1, 2, 3$$

$$y^{N_i} \in \{0,1\}^{|N_i|} \qquad \text{for } i = 1, 2, 3.$$

- Let $\{v^1, \ldots, v^p\} \subseteq \{0,1\}^{|N_1|+|N_2|+|N_3|}$ be all the vertices of the aggregated polyhedron.
- Notice that $p \leq (1 + |N_1|) \cdot (1 + |N_2|) \cdot (1 + |N_3|)$.

The aggregated polyhedron

$$\mu \sum_{k=1}^{|N_1|} k y^{N_1,k} + \lambda \sum_{k=1}^{|N_2|} k y^{N_2,k} + \sigma \sum_{k=1}^{|N_3|} k y^{N_3,k} \leq \beta$$

$$\sum_{k=1}^{|N_i|} y^{N_i,k} \leq 1 \qquad \text{for } i = 1, 2, 3$$

$$y^{N_i} \in \{0,1\}^{|N_i|} \qquad \text{for } i = 1, 2, 3.$$

- Let $\{v^1, \ldots, v^p\} \subseteq \{0,1\}^{|N_1|+|N_2|+|N_3|}$ be all the vertices of the aggregated polyhedron.
- Notice that $p \leq (1 + |N_1|) \cdot (1 + |N_2|) \cdot (1 + |N_3|)$.

The aggregated polyhedron

$$\mu \sum_{k=1}^{|N_1|} k y^{N_1,k} + \lambda \sum_{k=1}^{|N_2|} k y^{N_2,k} + \sigma \sum_{k=1}^{|N_3|} k y^{N_3,k} \leq \beta$$

$$\sum_{k=1}^{|N_i|} y^{N_i,k} \leq 1 \qquad \text{for } i = 1,2,3$$

$$y^{N_i} \in \{0,1\}^{|N_i|} \qquad \text{for } i = 1,2,3.$$

- Let $\{v^1, \ldots, v^p\} \subseteq \{0,1\}^{|N_1|+|N_2|+|N_3|}$ be all the vertices of the aggregated polyhedron.
- Notice that $p \leq (1 + |N_1|) \cdot (1 + |N_2|) \cdot (1 + |N_3|)$.

Theorem

The complete facet description in an extended space is:

$$y = \sum_{j=1}^{p} v^j z_j$$

$$\sum_{j=1}^{p} z_j = 1$$

$$z_j \geq 0 \qquad \text{for } j = 1, \ldots, p$$

$$\sum_{j \in N_i} x_j^{N_i} = \sum_{k=1}^{n_i} k y^{N_i,k} \qquad \text{for } i = 1,2,3$$

$$\sum_{j \in T} x_j^{N_i} \geq \sum_{\substack{k \in \{1,\ldots,n_i\}: \\ |T|+k > n_i}} (|T| + k - n_i) y^{N_i,k} \qquad \text{for } i = 1,2,3 \text{ and } \emptyset \neq T \subset N_i$$

$$x \in \mathbf{R}^{|N_1|+|N_2|+|N_3|}$$

$$y \in \mathbf{R}^{|N_1|+|N_2|+|N_3|}$$

$$z \in \mathbf{R}^p.$$

The simplification effect of branching

Initial Problem

2 constraints and 12 variables

13083 facets

1. Fix $x_2 = 0,\ x_6 = 0$
   690 facets
2. Fix $x_2 = 0,\ x_6 = 1$
   425 facets
3. Fix $x_2 = 1,\ x_6 = 0$
   91 facets
4. Fix $x_2 = 1,\ x_6 = 1$
   541 facets
5. Total : 1747 facets

Comparing Variable Branching with Value Disjunction

$\binom{12}{2}$ possible choices of x_i, x_j $\binom{12}{3}$ possible choices of x_r, x_s, x_t

Compute the number of facets for all four cases

$x_i = 0, x_j = 0,\ x_i = 1, x_j = 1$ $x_r + x_s + x_t = 0,\ x_r + x_s + x_t = 1$

$x_i = 1, x_j = 0,\ x_i = 0, x_j = 1$ $x_r + x_s + x_t = 2,\ x_r + x_s + x_t = 3$

Experiments with branching

The simplification effect of branching

Initial Problem

2 constraints and 12 variables

13083 facets

1. Fix $x_2 = 0,\ x_6 = 0$
   690 facets
2. Fix $x_2 = 0,\ x_6 = 1$
   425 facets
3. Fix $x_2 = 1,\ x_6 = 0$
   91 facets
4. Fix $x_2 = 1,\ x_6 = 1$
   541 facets
5. Total : 1747 facets

Comparing Variable Branching with Value Disjunction

$\binom{12}{2}$ possible choices of x_i, x_j $\quad$ $\binom{12}{3}$ possible choices of x_r, x_s, x_t

Compute the number of facets for all four cases

$x_i = 0, x_j = 0,\ x_i = 1, x_j = 1 \qquad x_r + x_s + x_t = 0,\ x_r + x_s + x_t = 1$

$x_i = 1, x_j = 0,\ x_i = 0, x_j = 1 \qquad x_r + x_s + x_t = 2,\ x_r + x_s + x_t = 3$

Experiments with branching

The simplification effect of branching

Initial Problem

2 constraints and 12 variables

13083 facets

1. Fix $x_2 = 0, \ x_6 = 0$
   690 facets
2. Fix $x_2 = 0, \ x_6 = 1$
   425 facets
3. Fix $x_2 = 1, \ x_6 = 0$
   91 facets
4. Fix $x_2 = 1, \ x_6 = 1$
   541 facets
5. Total : 1747 facets

Comparing Variable Branching with Value Disjunction

$\binom{12}{2}$ possible choices of x_i, x_j $\binom{12}{3}$ possible choices of x_r, x_s, x_t

Compute the number of facets for all four cases

$x_i = 0, x_j = 0, \ x_i = 1, x_j = 1$ $\quad x_r + x_s + x_t = 0, \ x_r + x_s + x_t = 1$

$x_i = 1, x_j = 0, \ x_i = 0, x_j = 1$ $\quad x_r + x_s + x_t = 2, \ x_r + x_s + x_t = 3$

Experiments with branching

The simplification effect of branching

Initial Problem

2 constraints and 12 variables

13083 facets

1. Fix $x_2 = 0, \ x_6 = 0$
   690 facets
2. Fix $x_2 = 0, \ x_6 = 1$
   425 facets
3. Fix $x_2 = 1, \ x_6 = 0$
   91 facets
4. Fix $x_2 = 1, \ x_6 = 1$
   541 facets
5. Total : 1747 facets

Comparing Variable Branching with Value Disjunction

$\binom{12}{2}$ possible choices of x_i, x_j $\binom{12}{3}$ possible choices of x_r, x_s, x_t

Compute the number of facets for all four cases

$x_i = 0, x_j = 0, \ x_i = 1, x_j = 1$ $x_r + x_s + x_t = 0, \ x_r + x_s + x_t = 1$

$x_i = 1, x_j = 0, \ x_i = 0, x_j = 1$ $x_r + x_s + x_t = 2, \ x_r + x_s + x_t = 3$

The simplification effect of branching

Initial Problem

2 constraints and 12 variables

13083 facets

1. Fix $x_2 = 0,\ x_6 = 0$
   690 facets
2. Fix $x_2 = 0,\ x_6 = 1$
   425 facets
3. Fix $x_2 = 1,\ x_6 = 0$
   91 facets
4. Fix $x_2 = 1,\ x_6 = 1$
   541 facets
5. Total : 1747 facets

Comparing Variable Branching with Value Disjunction

$\binom{12}{2}$ possible choices of x_i, x_j $\binom{12}{3}$ possible choices of x_r, x_s, x_t

Compute the number of facets for all four cases

$x_i = 0, x_j = 0,\ x_i = 1, x_j = 1$ $x_r + x_s + x_t = 0,\ x_r + x_s + x_t = 1$

$x_i = 1, x_j = 0,\ x_i = 0, x_j = 1$ $x_r + x_s + x_t = 2,\ x_r + x_s + x_t = 3$

Experiments with branching

The simplification effect of branching

Initial Problem

2 constraints and 12 variables

13083 facets

1. Fix $x_2 = 0,\ x_6 = 0$
   690 facets
2. Fix $x_2 = 0,\ x_6 = 1$
   425 facets
3. Fix $x_2 = 1,\ x_6 = 0$
   91 facets
4. Fix $x_2 = 1,\ x_6 = 1$
   541 facets
5. Total : 1747 facets

Comparing Variable Branching with Value Disjunction

$\binom{12}{2}$ possible choices of x_i, x_j $\binom{12}{3}$ possible choices of x_r, x_s, x_t

Compute the number of facets for all four cases

$x_i = 0, x_j = 0,\ x_i = 1, x_j = 1$ $x_r + x_s + x_t = 0,\ x_r + x_s + x_t = 1$

$x_i = 1, x_j = 0,\ x_i = 0, x_j = 1$ $x_r + x_s + x_t = 2,\ x_r + x_s + x_t = 3$

Experiments with branching

The simplification effect of branching

Initial Problem

2 constraints and 12 variables

13083 facets

1. Fix $x_2 = 0,\ x_6 = 0$
   690 facets
2. Fix $x_2 = 0,\ x_6 = 1$
   425 facets
3. Fix $x_2 = 1,\ x_6 = 0$
   91 facets
4. Fix $x_2 = 1,\ x_6 = 1$
   541 facets
5. Total : 1747 facets

Comparing Variable Branching with Value Disjunction

$\binom{12}{2}$ possible choices of x_i, x_j $\binom{12}{3}$ possible choices of x_r, x_s, x_t

Compute the number of facets for all four cases

$x_i = 0, x_j = 0,\ x_i = 1, x_j = 1$ $x_r + x_s + x_t = 0,\ x_r + x_s + x_t = 1$

$x_i = 1, x_j = 0,\ x_i = 0, x_j = 1$ $x_r + x_s + x_t = 2,\ x_r + x_s + x_t = 3$

Claim

It is efficient to use value disjunction on a set of variables that are similar (that have the same structure).

Ranking formula

We create a ranking formula that allows us to say whether a triple of variables is structured or not.

$$\begin{pmatrix} 7 & 8 & 7 \\ 11 & 9 & 10 \end{pmatrix} \text{ has a good ranking}$$

$$\begin{pmatrix} -23 & 12 & -6 \\ 4 & -1 & -14 \end{pmatrix} \text{ has a bad ranking}$$

Branching on value disjunctions vs. 2-variable branching

Claim

It is efficient to use value disjunction on a set of variables that are similar (that have the same structure).

Ranking formula

We create a ranking formula that allows us to say whether a triple of variables is structured or not.

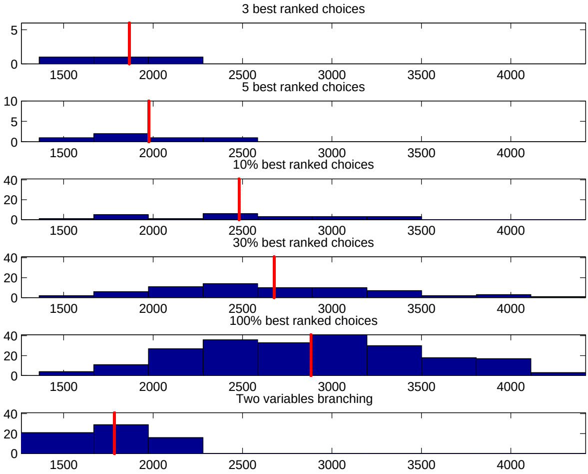
$$\begin{pmatrix} 7 & 8 & 7 \\ 11 & 9 & 10 \end{pmatrix} \text{ has a good ranking}$$

$$\begin{pmatrix} -23 & 12 & -6 \\ 4 & -1 & -14 \end{pmatrix} \text{ has a bad ranking}$$

Branching on value disjunctions vs. 2-variable branching ("unstructured")

Histograms of the total number of facets in the subproblems

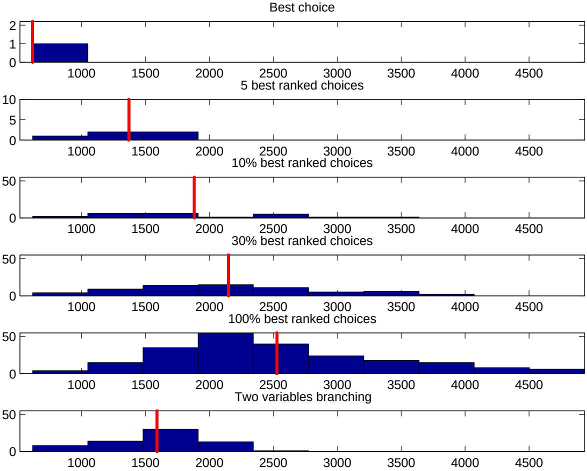
11	−7	9	10	−2	7	14	−15	4	−5	−2	−19	≤ 0
6	18	−4	−9	17	−11	5	−12	5	3	−18	7	≤ 0



Branching on value disjunctions vs. 2-variable branching ("structured")

Histograms of the total number of facets in the subproblems

$$\begin{array}{cccccccccccccc} 7 & 6 & 7 & 15 & -21 & -15 & -23 & -12 & 12 & -6 & 11 & 10 & \leq 0 \\ 10 & 10 & 9 & -21 & 4 & -3 & 4 & 13 & -1 & -14 & 2 & -6 & \leq 0 \end{array}$$



The market split problem

$$\begin{aligned} \min \quad & \sum_{i=1}^{m} |s_i| \\ \text{s. t.} \quad & \sum_{j=1}^{n} a_{ij} x_j + s_i = b \\ & x_j \in \{0,1\}. \end{aligned}$$

The value disjunction branching strategy

We suppose $a_{ij} \in [0, 100]$.
For each row i, we select all the variables $j \in T_i$ with $a_{ij} \geq 70$ and create m new rows

$$\sum_{j \in T} x_j$$

on which we branch simultaneously on the values.

Branching on market split

The market split problem

$$\begin{aligned} \min \ & \sum_{i=1}^{m} |s_i| \\ \text{s. t.} \ & \sum_{j=1}^{n} a_{ij} x_j + s_i = b \\ & x_j \in \{0, 1\}. \end{aligned}$$

The value disjunction branching strategy

We suppose $a_{ij} \in [0, 100]$.
For each row i, we select all the variables $j \in T_i$ with $a_{ij} \geq 70$ and create m new rows

$$\sum_{j \in T} x_j$$

on which we branch simultaneously on the values.

Branching for market split and mas instances

			CPLEX 9.1		Value Disjunctions	
Name	Rows	Cols	Nodes (10^6)	Time (s)	Nodes (10^6)	Time (s)
mspl535-1	5	35	13.8	2 431	3.8	809
mspl535-2	5	35	11.9	2 084	4.2	865
mspl535-3	5	35	17	2 946	9.8	1 970
mspl540-4	5	40	321	55 918	105	20 873
mspl540-5	5	40	231	39 787	87	17 267
mspl540-6	5	40	188	30 532	97	19 162
mspl650-7	6	50	***	***	20400	4.4 M
mas74	13	151	4.4	2463	1.2	1 194
mas76	12	151	0.667	289	0.063	35

Computation times in CPU seconds on a Sun Fire V890 with 1200 MHz UltraSPARC-IV processors