Integer Programming Methods
Week 3

Remy Spliet

Erasmus University Rotterdam

Dantzig-Wolfe reformulation

Lagrangian relaxation

- ILP:

$$\begin{aligned} \max \mathbf{c}^T\mathbf{x} & \\ A\mathbf{x} &\leq \mathbf{b} \\ D\mathbf{x} &\leq \mathbf{d} \\ \mathbf{x} &\in \mathbb{N}^n_+. \end{aligned}$$

- Lagrangian function:

$$\begin{aligned} \theta(\lambda) = \max \mathbf{c}^T\mathbf{x} &+ \lambda^T(\mathbf{d} - D\mathbf{x}) \\ A\mathbf{x} &\leq \mathbf{b} \\ \mathbf{x} &\in \mathbb{N}^n_+. \end{aligned}$$

✓ The Lagrangian relaxation is the upper bound

$$\min_{\lambda \geq 0} \theta(\lambda).$$

Lagrangian relaxation

Theorem

Denote by z_{IP}, z_{LR}, z_{LP} the optimal integer programming, Lagrangean relaxation and LP relaxation value, respectively. It holds that

$$z_{IP} \leq z_{LR} \leq z_{LP}.$$

- The value z_{LR} can be computed using a **subgradient algorithm**.

Dantzig-Wolfe relaxation

Theorem

Define

$$Q = \{\mathbf{x} \in \mathbb{N}^n_+ : A\mathbf{x} \leq \mathbf{b}\}.$$

Then

$$z_{LR} = \max\{\mathbf{c}^T\mathbf{x} : D\mathbf{x} \leq \mathbf{d}, \mathbf{x} \in conv(Q)\}.$$

- We can rewrite $\max\{\mathbf{c}^T\mathbf{x} : D\mathbf{x} \leq \mathbf{d}, \mathbf{x} \in \text{conv}(Q)\}$.
  - Let $\hat{\mathbf{x}}^1, \ldots, \hat{\mathbf{x}}^K$ be the extreme points of $\text{conv}(Q)$.
  - Let $\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^L$ be the extreme directions of $\text{conv}(Q)$.
- Any $\mathbf{x} \in \text{conv}(Q)$ can be written as

$$\mathbf{x} = \sum_{k=1}^{K} \hat{\mathbf{x}}^k \lambda^k + \sum_{l=1}^{L} \hat{\mathbf{y}}^l \mu^l$$

for $\lambda, \mu \geq \mathbf{0}$ and $\sum_{k=1}^{K} \lambda^k = 1$.

Dantzig-Wolfe relaxation

Definition

The **Dantzig-Wolfe** relaxation is

$$
\begin{aligned}
\max \sum_{k=1}^{K}(\mathbf{c}^T\hat{\mathbf{x}}^k)\lambda^k + \sum_{l=1}^{L}(\mathbf{c}^T\hat{\mathbf{y}}^l)\mu^l \\
\sum_{k=1}^{K}(D\hat{\mathbf{x}}^k)\lambda^k + \sum_{l=1}^{L}(D\hat{\mathbf{y}}^l)\mu^l &\leq \mathbf{d} \\
\sum_{k=1}^{K}\lambda^k &= 1 \\
\lambda^k &\in \mathbb{R}_+ \quad \forall k \in \{1,\dots,K\} \\
\mu^l &\in \mathbb{R}_+ \quad \forall l \in \{1,\dots,L\}.
\end{aligned}
$$

Dantzig-Wolfe relaxation

✓ The Dantzig-Wolfe relaxation is an LP problem.

x The Dantzig-Wolfe relaxation might have exponentially many variables.

Dantzig-Wolfe reformulation

Definition

The **Dantzig-Wolfe reformulation** of the original ILP is

$$
\begin{aligned}
\max \sum_{k=1}^{K}(\mathbf{c}^T\hat{\mathbf{x}}^k)\lambda^k + \sum_{l=1}^{L}(\mathbf{c}^T\hat{\mathbf{y}}^l)\mu^l & \\
\sum_{k=1}^{K}(D\hat{\mathbf{x}}^k)\lambda^k + \sum_{l=1}^{L}(D\hat{\mathbf{y}}^l)\mu^l & \leq \mathbf{d} \\
\sum_{k=1}^{K}\lambda^k & = 1 \\
\lambda^k & \in \mathbb{R}_+ \quad \forall k \in \{1,\ldots,K\} \\
\mu^l & \in \mathbb{R}_+ \quad \forall l \in \{1,\ldots,L\} \\
\sum_{k=1}^{K}\hat{\mathbf{x}}^k\lambda^k + \sum_{l=1}^{L}\hat{\mathbf{y}}^l\mu^l & \in \mathbb{N}_+^n.
\end{aligned}
$$

Dantzig-Wolfe decomposition

$$
\begin{array}{lllllllllll}
\max & \mathbf{c}_1^T\mathbf{x}_1 & + & \mathbf{c}_2^T\mathbf{x}_2 & + & \dots & + & \mathbf{c}_n^T\mathbf{x}_n & & \\
 & D_1\mathbf{x}_1 & + & D_2\mathbf{x}_2 & + & \dots & + & D_n\mathbf{x}_n & \leq & \mathbf{d} \\
 & A_1\mathbf{x}_1 & & & & & & & \leq & \mathbf{b}_1 \\
 & & & A_2\mathbf{x}_2 & & & & & \leq & \mathbf{b}_2 \\
 & & & & & \ddots & & & \vdots & \\
 & & & & & & & A_n\mathbf{x}_n & \leq & \mathbf{b}_n \\
 & \mathbf{x}_1 & , & \mathbf{x}_2 & , & \dots & , & \mathbf{x}_n & \in & \mathbb{N}_+^M
\end{array}
$$

An ILP problem of the above from is said to have a **block diagonal structure**.

Reformulation

- **Dantzig-Wolfe decomposition** considers each block separately.
- Consider for each block $j = 1, \dots, n$ the auxiliary set

$$Q_j = \{\mathbf{x} \in \mathbb{N}_+^{m_j} | A_j \mathbf{x} \leq \mathbf{b}_j\}$$

- Let $\hat{\mathbf{x}}_j^k$ for $k = 1, \dots, K_j$ be the extreme points of $\text{conv}(Q_j)$.
- Let $\hat{\mathbf{y}}_j^l$ for $l = 1, \dots, L_j$ be the extreme directions of $\text{conv}(Q_j)$.

- Any $\mathbf{x} \in \text{conv}(Q_j)$ can be written as

$$\mathbf{x} = \sum_{k=1}^{K_j} \hat{\mathbf{x}}_j^k \lambda_j^k + \sum_{l=1}^{L_j} \hat{\mathbf{y}}_j^l \mu_j^l$$

for $\lambda_j, \mu_j \geq \mathbf{0}$ and $\sum_{k=1}^{K_j} \lambda_j^k = 1$.

Dantzig-Wolfe decomposition

The Dantzig-Wolfe relaxation is:

$$\max \sum_{k=1}^{K_1}(\mathbf{c}_1^T\hat{\mathbf{x}}_1^k)\lambda_1^k+\sum_{l=1}^{L_1}(\mathbf{c}_1^T\hat{\mathbf{y}}_1^l)\mu_1^l+\ldots+\sum_{k=1}^{K_n}(\mathbf{c}_n^T\hat{\mathbf{x}}_n^k)\lambda_n^k+\sum_{l=1}^{L_n}(\mathbf{c}_n^T\hat{\mathbf{y}}_n^l)\mu_n^l$$

$$\sum_{k=1}^{K_1}(D_1\hat{\mathbf{x}}_1^k)\lambda_1^k+\sum_{l=1}^{L_1}(D_1\hat{\mathbf{y}}_1^l)\mu_1^l \quad + \ \ldots$$

$$\ldots + \sum_{k=1}^{K_n}(D_n\hat{\mathbf{x}}_n^k)\lambda_n^k+\sum_{l=1}^{L_n}(D_n\hat{\mathbf{y}}_n^l)\mu_n^l \leq \mathbf{d}$$

$$\sum_{k=1}^{K_1}\lambda_1^k = 1$$

$$\ddots \qquad \vdots$$

$$\sum_{k=1}^{K_n}\lambda_n^k = 1$$

$$\lambda_j^k \geq 0, \ k=1, \ldots, K_j, \ \mu_j^l \geq 0, \quad l=1,\ldots,L_j$$

Column generation

Column generation

We consider a (maximizing) LP with a very many variables:

- ▶ We use the simplex method.
- x There are very many non-basic variables.
- x Computing the reduced costs of **all** non-basic variables takes much time.
- ▶ In **column generation**, we only include a **few** variables.
- ▶ **Generate** a variable (i.e., column) with a positive reduced cost without computing all reduced costs.
- ✓ This way the LP can be solved **faster**.

Column generation

- Consider n variables and let $N = \{1, \ldots, n\}$.
- Consider m constraints and let $M = \{1, \ldots, m\}$.
- Consider the following LP.

$$
\begin{aligned}
\max \sum_{i \in N} c_i x_i & \\
\sum_{i \in N} a_{ij} x_i &= b_j \qquad \forall j \in M \\
x_i &\geq 0 \qquad \forall i \in N
\end{aligned}
$$

- In column generation, the LP to be solved is also called the **master problem** (MP).
- Denote the optimal solution value by z_{MP}.

Column generation:

Restricted master problem

Restricted master problem

Restricted master problem (RMP):
The master problem in which only a subset $N' \subset N$ of the variables is used.

$$
\begin{aligned}
z_{RMP} = \max \sum_{i \in N'} c_i x_i & \\
\sum_{i \in N'} a_{ij} x_i &= b_j \quad \forall j \in M \\
x_i &\geq 0 \quad \forall i \in N'
\end{aligned}
$$

Restricted master problem

- The column generation algorithm **initializes** by choosing N'.
- Compute z_{RMP}.
- Because not all variables are included $z_{RMP} \leq z_{MP}$.
- Variables are iteratively added to N' to **increase** z_{RMP} (or at least not decrease z_{RMP}).
- The column generation algorithm **terminates** when $z_{RMP} = z_{MP}$.

? How do we know which variables to add?

? How do we know if $z_{RMP} = z_{MP}$?

Column generation:

Pricing problem

Pricing problem

Precisely as in the simplex method...

- Look for a non-basic variable with positive reduced costs.
- Let λ_j, $j \in M$, be dual variables corresponding to the constraints of

$$\begin{aligned} z_{RMP} = \max \sum_{i \in N'} c_i x_i & & \\ \sum_{i \in N'} a_{ij} x_i &= b_j & \forall j \in M \\ x_i &\geq 0 & \forall i \in N'. \end{aligned}$$

- Consider an optimal basis of RMP and optimal duals.
- The reduced cost of a variable x_i is

$$RC(x_i) = c_i - \sum_{j \in M} \lambda_j a_{ij}$$

Pricing problem

- **x** Computing the reduced cost of all non-basic variables takes time.

- **?** Can we avoid computing all reduced costs?

Pricing problem

Pricing problem:

$$\max_{i \in N} RC(x_i).$$

- If the maximum is greater than 0:
  - A variable with a positive reduced cost is found.
  - **Add** this variable to the RMP.
- Otherwise, no positive reduced cost variable exists.
  - The current solution to RMP is also optimal for MP (so $z_{RMP} = z_{MP}$).
  - **Terminate** the column generation algorithm.

Pricing problem

Pricing algorithm:
An algorithm used to solve the pricing problem.

? How do we solve the pricing problem?

✓ Generic pricing algorithm:

- Make a list of all variables.
- Compute all reduced costs.
- Pick the highest reduced costs.

x This is precisely the simplex method.

✓ In many special cases, **special purpose pricing algorithms** are used.

! Crucial steps for developing a column generation algorithm:

- Modeling the pricing problem.
- Developing a pricing algorithm.

Summary of the column generation algorithm

Step 1 Initialize **restricted master problem**.

Step 2 Solve RMP.

Step 3 Solve **pricing problem**.

Step 4
- If a variable with **positive reduced costs** is found, add the variable with the highest reduced costs and return to Step 2.
- Otherwise, stop.

Column generation:

Computational considerations

Initialization

? How do we select an initial set of variables?

! The initial variables should provide a feasible solution.

✓ An **artificial variable** can be used.

- Coefficients such that all constraints are satisfied.
- Very high costs such that it is never part of an optimal solution.

x An artificial variable provides a poor initial solution.

- Column generation can take long.
- The algorithm can become numerically unstable.

✓ In practice, to come up with initial variables

- Use **expert knowledge**,
- Use a **heuristic**.

Pricing heuristics

! The pricing problem is crucial for the success of column generation:

- The pricing problem might be **hard** to solve.
- The pricing algorithm might take **much time**.

? What to do if the pricing algorithm takes (too) much time?

Pricing heuristics

Pricing heuristic:
A pricing algorithm that does not guarantee optimality.

✓ If the pricing heuristic **succesfully finds** a positive reduced cost variable, add it.

x If the pricing heuristic **fails to find** a positive reduced cost variable, one might still exist.

? What use is a pricing heuristic?

Pricing heuristics

- **Accelerate the pricing algorithm:**
  - Step 1 Use pricing heuristic.
  - Step 2 If a positive reduced cost column is found, add it and stop. Otherwise continue with step 3.
  - Step 3 Use an exact pricing algorithm.
- **Ideally, Step 3 is executed once**, to verify that no positive reduced cost variable exists.
- Multiple pricing heuristics could be used sequentially.

Pricing heuristics

Column generation heuristic:
A column generation algorithm in which the pricing algorithm is a pricing heuristic.

- If the pricing heuristic fails to find a positive reduced cost variable, the column generation heuristic **terminates**.

x So a column generation heuristic might stop too soon.

✓ Hence, a **lower bound** on the maximum LP value is found.

!!! When solving an LP relaxation of an ILP, be careful!

Column management

- ✗ The number of columns in the RMP can grow too large.
- We can manage the columns in the RMP.

Examples of column management:

- Remove every column that was not part of an RMP solution for X iterations.
- Remove every column with reduced cost smaller than α, with $\alpha \leq 0$.

Column pool

- Removed columns can be stored in a **column pool**.
- As a pricing heuristic, we can check the columns in the column pool.

Branch-and-price

Branch-and-price
A branch-and-bound algorithm in which LP relaxations are solved using column generation.

✓ Branch-and-price solves an ILP with many variables.

! A **branching rule** for branch-and-price requires extra care:

- The LP relaxation is affected.
- The pricing problem is affected.
- A different pricing algorithm might be needed.

x It is usually a bad idea to branch on the variables directly.

- Avoid changes of the pricing problem when branching, so the same pricing algorithm can be used in all nodes of the branching tree.

Benders decomposition

Benders decomposition

We consider a mathematical programming problem using the below notation:

- $\mathbf{x} \in \mathbb{R}_+^n$ are continuous decision variables.
- $\mathbf{y}$ are decision variables taking values in the **closed and bounded** set S.
  - For example $S = \{0,1\}^k$ such that y are binary variables.
- $c \in \mathbb{R}^n$ are the linear cost coefficients of $\mathbf{x}$.
- $f : S \to \mathbb{R}$ is the cost function of y.
  - For example f is the quadratic function $f(\mathbf{y}) = \mathbf{y}^T\mathbf{y}$.
- $A \in \mathbb{R}^{m\times n}$ is the restriction matrix corresponding to $\mathbf{x}$.
- $g : S \to \mathbb{R}^m$ is the restriction function of $\mathbf{y}$.
  - For example g is a quadratic function using the $m \times n$ matrix G: $g(\mathbf{y}) = \mathbf{y}^T G\mathbf{y}$.
- $b \in \mathbb{R}^m$ are the restricting values.

Benders decomposition

$$
\begin{array}{rrrcl}
\min & \mathbf{c}^T\mathbf{x} & + \; f(\mathbf{y}) & & \\
 & A\mathbf{x} & + \; g(\mathbf{y}) & \geq & \mathbf{b} \\
 & \mathbf{x} & & \geq & \mathbf{0} \\
 & & \mathbf{y} & \in & S
\end{array}
$$

Benders decomposition

Benders decomposition decomposes the "easy" **x**-part and the "difficult" **y**-part of the problem.

- Benders **originally** introduced his decomposition to **solve MILPs**.
  - Benders, J. F. 1962 "Partitioning procedures for solving mixed-variables programming problems", *Numerische Mathematik*, Vol. 4, No. 1 , pp. 238-252.
- Benders decomposition is **later generalized** to possibly nonlinear functions f, g and arbitrary feasibility sets S.
- Reformulate the problem.
  - Optimize **x** for fixed **y**.
  - Construct LP dual for fixed **y**.
  - Express in terms of extreme points and directions.

Benders decomposition:

Which y values are really feasible?

Which $\mathbf{y}$ values are really feasible?

! Reformulation hinges on a good representation of the feasible region of the $\mathbf{y}$ variables.

✓ Farkas' Lemma is the key.

Lemma

(One of the variants of Farkas' Lemma.)
There exists a vector $\mathbf{x} \geq \mathbf{0}$ *such that* $P\mathbf{x} \geq \mathbf{q}$ *if and only if* $\mathbf{q}^T\mathbf{u} \leq 0$ *for all* $\mathbf{u}$ *satisfying* $P^T\mathbf{u} \leq \mathbf{0}$ *and* $\mathbf{u} \geq \mathbf{0}$.

Which $\mathbf{y}$ values are really feasible?

- Let R be the set of $\mathbf{y}$ values that are really feasible:

  $$R = \{\mathbf{y} \in S | \text{ there exists an } \mathbf{x} \geq \mathbf{0} \text{ such that } A\mathbf{x} \geq \mathbf{b} - g(\mathbf{y}) \ \}$$

- We apply Farkas' Lemma with $P = A$, $\mathbf{q} = (\mathbf{b} - g(\mathbf{y}))$

  $$\begin{aligned} R &= \{\mathbf{y} \in S | \ \text{there exists an } \mathbf{x} \geq \mathbf{0} \text{ such that } A\mathbf{x} \geq \mathbf{b} - g(\mathbf{y}) \ \} \\ &= \{\mathbf{y} \in S | \ (\mathbf{b} - g(\mathbf{y}))^T \mathbf{u} \leq 0 \\ &\qquad\qquad \text{for all } \mathbf{u} \text{ satisfying } A^T\mathbf{u} \leq \mathbf{0} \text{ and } \mathbf{u} \geq \mathbf{0} \ \} \end{aligned}$$

Which $\mathbf{y}$ values are really feasible?

- Consider the polyhedral cone

$$C = \{\mathbf{u} | A^T\mathbf{u} \leq \mathbf{0},\ \mathbf{u} \geq \mathbf{0}\}$$

- We know that a polyhedral cone has one extreme point $\hat{\mathbf{u}} = \mathbf{0}$ and a finite number n_c of extreme directions $\hat{\mathbf{v}}_j^c$ for $j = 1, \ldots, n_c$.

- We can write every $\mathbf{u} \in C$ as

$$\mathbf{u} = \sum_{j=1}^{n_c} \lambda_j \hat{\mathbf{v}}_j^c \qquad \lambda_j \geq 0,\ j = 1, \ldots, n_c$$

- We can rewrite R as

$$R \ = \{\mathbf{y} \in S |\ (\mathbf{b} - g(\mathbf{y}))^T \sum_{j=1}^{n_c} \lambda_j \hat{\mathbf{v}}_j^c \leq 0 \ \text{for all } \lambda_j \geq 0,\ j = 1, \ldots, n_c\}$$

Which $\mathbf{y}$ values are really feasible?

- The requirement $(\mathbf{b} - g(\mathbf{y}))^T \sum_{j=1}^{n_c} \lambda_j \hat{\mathbf{v}}_j^c \leq 0$ for **all** $\lambda_j \geq 0$ holds if and only if

$$(\mathbf{b} - g(\mathbf{y}))^T \hat{\mathbf{v}}_j^c \leq 0 \qquad \text{for all } j = 1, \ldots, n_c$$

- Hence, we can rewrite R into its final version

$$R = \{\mathbf{y} \in S | \ (\mathbf{b} - g(\mathbf{y}))^T \hat{\mathbf{v}}_j^c \leq 0, \ \text{for all } j = 1, \ldots, n_c\}$$

Benders decomposition:

Reformulation

Reformulation

- We rewrite the problem as a nested optimization problem

$$\min_{\mathbf{y} \in R} \left\{ f(\mathbf{y}) + \min\{\mathbf{c}^T\mathbf{x} | A\mathbf{x} \geq \mathbf{b} - g(\mathbf{y}), \mathbf{x} \geq 0\} \right\}$$

- The inner optimization problem corresponds to optimizing **x** for fixed **y**.
- Next we
  - rewrite the inner optimization problem using LP duality,
  - express the variables in terms of extreme points and directions.

Reformulation

$$\min_{\mathbf{y}\in R}\{f(\mathbf{y})+\min\{\mathbf{c}^T\mathbf{x}|A\mathbf{x}\geq \mathbf{b}-g(\mathbf{y}),\mathbf{x}\geq 0\}\}$$

$$=\min_{\mathbf{y}\in R}\{f(\mathbf{y})+\max\{(\mathbf{b}-g(\mathbf{y}))^T\mathbf{u}|A^T\mathbf{u}\leq \mathbf{c},\mathbf{u}\geq 0\}\}$$

$$=\min_{\mathbf{y}\in R}\{f(\mathbf{y})+\max_{1\leq i\leq n_p}\{(\mathbf{b}-g(\mathbf{y}))^T\hat{\mathbf{u}}_i^p\}\}$$

- Here $\hat{\mathbf{u}}_i^p$ are the extreme points of

$$P=\{\mathbf{u}|A^T\mathbf{u}\leq \mathbf{c},\mathbf{u}\geq \mathbf{0}\}$$

- Observe that the final step is valid since $y\in R$ ensures that the inner optimization problem is feasible, hence the dual cannot be unbounded.

Reformulation

$$\min_{\mathbf{y}\in R}\{f(\mathbf{y}) + \max_{1\le i\le n_p}\{(\mathbf{b}-g(\mathbf{y}))^T\hat{\mathbf{u}}_i^p\}\}$$

$$\begin{aligned} = \min\ & z \\ & z \ge f(\mathbf{y}) + (\mathbf{b}-g(\mathbf{y}))^T\hat{\mathbf{u}}_i^p \qquad i=1,\ldots,n_p \\ & \mathbf{y}\in R \end{aligned}$$

$$\begin{aligned} = \min\ & z \\ & z \ge f(\mathbf{y}) + (\mathbf{b}-g(\mathbf{y}))^T\hat{\mathbf{u}}_i^p \qquad i=1,\ldots,n_p \\ & 0 \ge (\mathbf{b}-g(\mathbf{y}))^T\hat{\mathbf{v}}_j^c \qquad j=1,\ldots,n_c \\ & \mathbf{y}\in S \end{aligned}$$

Reformulation

$$
\begin{aligned}
\min \ & z \\
& z \geq f(\mathbf{y}) + (\mathbf{b} - g(\mathbf{y}))^T \hat{\mathbf{u}}_i^p \qquad i = 1, \ldots, n_p \\
& 0 \geq \qquad\quad (\mathbf{b} - g(\mathbf{y}))^T \hat{\mathbf{v}}_j^c \qquad j = 1, \ldots, n_c \\
& \mathbf{y} \in S
\end{aligned}
$$

Reformulation

- ✓ Observe that $C = \text{rec}(P)$.
- **!** Hence, the formulation contains a constraint for every extreme point and direction of P.
- **x** The formulation might have exponentially many constraints.

Row generation

Row generation

- Suppose the LP has **many constraints** (instead of variables).

! The **dual** problem is an LP with **many variables**.

✓ We can apply column generation on the dual problem.

- This is effectively **row generation** on the primal problem.