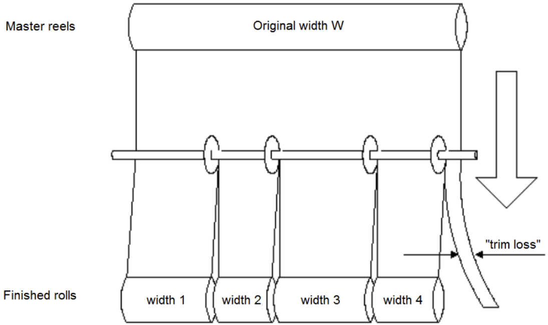
Lecture 22 Column Generation and Dantzig Wolfe Decomposition

Can Li

ChE 597: Computational Optimization
Purdue University

The cutting stock problem

- Consider a paper mill that has a number of rolls of paper of fixed width. Customers demand different numbers of rolls of various-sized widths.



The Cutting Stock Problem

Consider a paper mill that has a number of rolls of paper of fixed width. Customers demand different numbers of rolls of various-sized widths.

- Given $K = 20$ rolls of width $W = 100$ inches.
- Widths w_i (inches) and Demand n_i (rolls) are given as:

Width w_i (inches)	25	35	40
Demand n_i (rolls)	7	5	3

For example, a master roll can be cut into the following patterns:

- 4 rolls each of width 25 inches.
- 2 rolls of width 35 + 1 roll of width 25 (resulting in a waste of 5)

Determine a cutting plan that:

- Satisfies all demands
- Minimizes number of used rolls

Question: How to formulate the optimization problem?

The Cutting Stock Problem: Classical Formulation

Originally proposed by Kantorovich in "Mathematical methods of planning and organizing production" (1939 in Russian, 1960 in English). Economics Nobel, 1975.

- Decision variables

$$y_k = \begin{cases} 1 & \text{if master roll } k \text{ is cut} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ik} = \text{number of rolls of width } w_i \text{ cut on master roll } k$$

- Objective: minimize number of rolls used $\sum_{k=1}^{K} y_k$
- Constraints
  - Satisfy demand of each type of roll: $\sum_{k=1}^{K} z_{ik} \geq n_i$ for $i = 1, \ldots, m$.
  - Cut no more than available: $\sum_{i=1}^{m} w_i z_{ik} \leq W y_k$ for $k = 1, \ldots, K$.

The Cutting Stock Problem: Classical Formulation

- Decision variables:

$$y_k = \begin{cases} 1 & \text{if master roll } k \text{ is cut} \\ 0 & \text{otherwise} \end{cases}$$

$$z_{ik} = \text{number of rolls of width } w_i \text{ cut on master roll } k$$

- Objective:

$$\min \quad \sum_{k=1}^{K} y_k$$

- Subject to:

$$\sum_{k=1}^{K} z_{ik} \geq n_i \quad \forall i = 1, \ldots, m$$

$$\sum_{i=1}^{m} w_i z_{ik} \leq W y_k \quad \forall k = 1, \ldots, K$$

$$y_k \in \{0, 1\} \quad \forall k = 1, \ldots, K$$

$$z_{ik} \in \mathbb{Z}_+ \quad \forall k = 1, \ldots, K, \forall i = 1, \ldots, m$$

The Cutting Stock Problem: Classical Formulation

How good is this model?

- This formulation has poor computational performance in practice.
- Theoretical justifications for poor performance?
  - Weak LP relaxation
  - Highly symmetric/degenerate
- Question: How to break symmetry?
- Is there an alternative formulation?

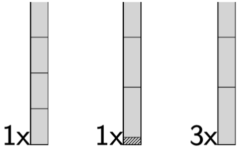
The Cutting Stock Problem: Set Covering Formulation

Originally proposed by Gilmore and Gomory in "A linear programming approach to the cutting-stock problem", Operations Research (1961).

- Key observation: Optimal solution uses only a few of all possible cutting patterns.

Width w_i	Patterns p			Demand n_i
25	4	1	1	7
35	0	2	1	5
40	0	0	1	3

Optimal solution uses only 3/15 possible patterns.



The Cutting Stock Problem: Set Covering Formulation

- Key observation: Optimal solution uses only a few of all possible cutting patterns.

Width w_i	Patterns p			Demand n_i
25	$4x_1$	$+1x_2$	$+1x_3$	≥ 7
35	$0x_1$	$+2x_2$	$+1x_3$	≥ 5
40	$0x_1$	$+0x_2$	$+1x_3$	≥ 3

- Decision variables
- $x_p =$ number of master rolls to be cut using pattern p
- Objective: minimize number of rolls used $\sum_p x_p$

The Cutting Stock Problem: Set Covering Formulation

$$\min_x \quad \sum_{p \in P} x_p$$

$$\text{s.t.} \sum_{p \in P} a_{ip} x_p \geq n_i \quad \forall i = 1, \ldots, m$$

$$x_p \in \mathbb{Z}_+ \quad \forall p \in P \quad (=\text{feasible patterns})$$

- a_{ip} = number of rolls of width w_i in pattern p
- **Question:** How many feasible patterns are possible?
- $\approx \binom{m}{j}$, where j = average no. of cuts in a feasible pattern.
- j can be estimated as $W/\bar{w}$ where $\bar{w}$ = average width $= \frac{\sum_i w_i n_i}{\sum_i n_i}$.
- This is combinatorial growth. For example, a typical real-world problem contains about $m = 50$ orders (widths). If there are $j = 10$ average cuts per master roll, then the number of feasible patterns is more than 10 billion.

The Cutting Stock Problem: Set Covering Formulation

How good is this model?

- Theoretical properties
  - No symmetry issues
  - Strong LP relaxations

Computational properties

- If we explicitly list all the patterns, we would run out of memory.
- If we had the "optimal patterns" from the start, we could easily solve the problem. But we don't know which ones these are.
- We know that the vast majority of patterns are useless.
  - Question: How do we find the useful patterns?

Properties of the LP relaxation

Relax the integer variables x_p, $p \in P$ as continuous variables.

$$\min_x \quad \sum_{p \in P} x_p$$

$$\text{s.t.} \sum_{p \in P} a_{ip} x_p = n_i \quad \forall i = 1, \ldots, m \quad \text{(assume equality for simplicity)}$$

$$x_p \geq 0 \quad \forall p \in P \quad \text{(=feasible patterns)}$$

- The optimal solution is always obtained at a basic feasible solution (BFS).
- At a BFS, out of the $|P|$ inequalities constraints, $|P| - m$ of them are active. At least $|P| - m$ of the constraints $x_p \geq 0$ are active.
- At a BFS solution, at most m $x_p > 0$. In other words, at most m patterns are used in the optimal solution.
- We only need the columns of a_{ip} correspond to the optimal patterns.
- If we round up every non-zero LP variable to the nearest integer, this integer solution is feasible and has value at most m more than LP solution.

Column generation

Basic idea:

- Start with an initial subset of feasible patterns
- Solve the LP relaxation assuming these are the only feasible patterns
- Check if the inclusion of any pattern that has been left out can improve the objective (*"pricing"*) $\rightarrow$ the most critical step
- Iterate

Start with an Initial Basic Feasible Solution

Start with a small subset of patterns $P^0 \subseteq P$,

$$\min_x \quad \sum_{p \in P^0} x_p$$

$$\text{s.t.} \sum_{p \in P^0} a_{ip} x_p = n_i \quad \forall i = 1, \ldots, m$$

$$x_p \geq 0 \quad \forall p \in P^0 \quad (\text{=a subset of all the feasible patterns})$$

- For example, P^0 can have only m patterns just to ensure that the problem is feasible. Each width is covered by at least one pattern. Finding an initial basic feasible solution is easy for this problem. For $i = 1, \ldots, m$, we may let the i th pattern consist of one roll of width w_j and none of the other widths. Then, the m columns of A form a feasible basis.
- A BFS solution to this is problem is also a BFS to the original problem with $|P|$ patterns. We can just set the rest of the variables in $P \backslash P^0$ to zero (nonbasic).
- Recall in the primal simplex method, the optimality condition is the reduced costs of the nonbasic variables being nonnegative.

Computing Reduced Costs

- Suppose the optimal basis matrix of the problem being B
- Dual multipliers $y^T = c_B^T (B)^{-1}$
- Reduced costs of pattern p: $\bar{c}_p = c_p - A_p^T y$.
- Note that $c_p = 1$ for all p (each pattern consumes one roll of paper).
- If $A_p^T y > 1$ for any p, we have a column with a negative reduced cost.
- The next step is to find a column (pattern) with negative reduced cost.

Integer Programming for Negative Reduced Cost

$$\begin{aligned} \max_{a} \quad & \sum_{i=1}^{m} y_i a_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} w_i a_i \leq W \\ & a_i \in \mathbb{Z}^+ \end{aligned}$$

- Solving this IP yields the column with the least reduced cost.
- If the optimal objective value is less than 1, no more columns with negative reduced costs exist.
- Otherwise, the optimal a gives the new pattern to enter the basis.
- The pricing problem can also be seen as finding the "most violated" constraints of the dual problem.
- This IP is the Knapsack Problem, solvable via dynamic programming.

Applications of Column Generation

- Cutting stock problem
- Vehicle routing problem
- Aircraft routing
- Crew scheduling

LP Problem Structure

- Consider an LP problem with two sets of decision variables x_1 and x_2.
- Variables x_1 and x_2 are subject to their own constraints (m_1 and m_2 constraints, respectively) and m_0 shared coupling constraints.
- Matrices D_1, D_2, F_1, F_2 define the system.

$$
\begin{aligned}
\min \quad & c_1^T x_1 + c_2^T x_2 \\
\text{s.t.} \quad & D_1 x_1 + D_2 x_2 = b_0 \\
& F_1 x_1 = b_1 \\
& F_2 x_2 = b_2 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Reformulation Using Minkowski-Weyl Theorem

$$\begin{aligned} \min \quad & c_1^T x_1 + c_2^T x_2 \\ \text{s.t.} \quad & D_1 x_1 + D_2 x_2 = b_0 \\ & x_1 \in P_1, x_2 \in P_2 \end{aligned}$$

where $P_i = \{x_i, F_i x_i = b_i, x_i \geq 0\}$

- For $i = 1, 2$, let x_i^j be the extreme points and w_i^k be extreme rays of P_i.
- Any $x_i \in P_i$ can be represented by a convex combination of the extreme points and conic combination of the extreme rays of P_i.

$$x_i = \sum_{j \in J_i} \lambda_i^j x_i^j + \sum_{k \in K_i} \theta_i^k w_i^k$$

$$\sum_{j \in J_i} \lambda_i^j = 1 \quad \text{for } i = 1, 2$$

$$\lambda_i^j \geq 0, \quad \forall i = 1, 2, \ j \in J_i \quad \theta_i^k \geq 0 \quad \text{for } i = 1, 2, k \in K_i$$

Master Problem Formulation

- The master problem is a reformulation with decision variables λ_i^j and θ_i^k.
- $m_0 + 2$ equality constraints. At a BFS, we can have at most $m_0 + 2$ λ_i^j and θ_i^k being nonzero.

$$
\begin{aligned}
\min \sum_{j \in J_1} \lambda_1^j c_1^T x_1^j + \sum_{k \in K_1} \theta_1^k c_1^T w_1^k + \sum_{j \in J_2} \lambda_2^j c_2^T x_2^j + \sum_{k \in K_2} \theta_2^k c_2^T w_2^k \\
\text{s.t.} \sum_{j \in J_1} \lambda_1^j D_1 x_1^j + \sum_{k \in K_1} \theta_1^k D_1 w_1^k + \sum_{j \in J_2} \lambda_2^j D_2 x_2^j + \sum_{k \in K_2} \theta_2^k D_2 w_2^k = b_0 \\
\sum_{j \in J_i} \lambda_i^j = 1 \quad i = 1, 2 \\
\lambda_i^j \geq 0, \quad \forall i = 1, 2,\ j \in J_i \quad \theta_i^k \geq 0 \quad \forall i = 1, 2,\ k \in K_i
\end{aligned}
$$

Master Problem vs. Original Problem

- Original problem has $m_0 + m_1 + m_2$ equality constraints.
- Master problem simplifies to $m_0 + 2$ equality constraints.
- Decision variables in the master problem could be very large in number because of the large number of extreme points and extreme rays.
- At a BFS, we can have at most $m_0 + 2$ λ_i^j and θ_i^k being nonzero. Most of the variables are zero (nonbasic).
- Solution: use column generation to generate the columns corresponding to the basic variables.

Basis Matrix and Dual Vector

- Start with a master problem with only a small subset of extreme points and extreme rays.
- Consider basis matrix B and its inverse B^{-1}.
- Dual vector $p^T = c_B^T B^{-1}$.
- Vector p has $m_0 + 2$ components: q for the first m_0 and r_1, r_2 for the last two.
- Recall the formula for the reduced cost for the jth variable is $c_j - A_j^T p$

Dantzig Wolfe Subproblem

- Reduced cost of λ_1^j: $(c_1^T - q^T D_1)x_1^j - r_1$.
- Reduced cost of θ_1^k: $(c_1^T - q^T D_1)w_1^k$.
- Objective is to identify if any reduced costs are negative.

Form the subproblem to decide on optimality:

$$\begin{aligned} \min \quad & (c_1^T - q^T D_1)x_1 \\ \text{s.t.} \quad & x_1 \in P_1 \end{aligned}$$

- Solve using simplex method.
- If optimal cost is $-\infty$, we find an extreme ray w_1^k that $(c_1^T - q^T D_1)w_1^k < 0$.
- If the optimal cost is finite and smaller than r_1, we find an extreme point x_1^j with negative reduced cost, i.e., $(c_1^T - q^T D_1)x_1^j - r_1 < 0$.
- If the the optimal cost is finite and no smaller than r_1. $(c_1^T - q^T D_1)w_1^k \geq 0$ for all extreme rays and $(c_1^T - q^T D_1)x_1^j - r_1 \geq 0$ for all extreme points. We can terminate.

Decomposition Algorithm - Iteration Steps

1. Start with a BFS to the master problem.
2. Solve two subproblems for P_1 and P_2.
   - If both subproblems yield nonnegative reduced costs, current solution is optimal.
   - If not, determine the entering variable based on the subproblem with a negative reduced cost. Add columns corresponding to the extreme point or the extreme ray that yield the negative reduced cost.
3. Update the master problem and repeat.

Applicability to multiple subproblems

$$\begin{aligned} \min \quad & c_1^T x_1 + c_2^T x_2 + \cdots + c_t^T x_t \\ \text{s.t.} \quad & D_1 x_1 + D_2 x_2 + \cdots + D_t x_t = b_0 \\ & F_i x_i = b_i, \qquad\qquad\qquad\qquad i = 1, 2, \ldots, t \\ & x_1, x_2, \ldots, x_t \geq 0 \end{aligned}$$

The only difference is that at each iteration of the algorithm, we may have to solve t subproblems.

References

- Bertsimas, D., & Tsitsiklis, J. N. (1997). Introduction to linear optimization. Belmont, MA: Athena scientific.