Large-scale optimization and decomposition methods: outline

- Solution approaches for large-scaled problems:
  - Delayed column generation
  - Cutting plane methods (delayed constraint generation)[7]
- Problems amenable to the above methods:
  - Cutting stock problem, etc.
  - Problems reformulated via decomposition methods
    - Benders decomposition
    - Dantzig-Wolfe decomposition

[7] Not to be confused with the "cutting stock problem," which is actually solved with a delayed column generation algorithm

Column Generation and Cutting Plane methods: a unified view

$$
\begin{array}{llll}
(P) & \min & \mathbf{c}^T\mathbf{x} \\
 & \text{s.t.} & \mathbf{A}\mathbf{x} = \mathbf{b} \\
 & & \mathbf{x} \geq \mathbf{0}
\end{array}
\qquad
\begin{array}{lll}
(D) & \max & \mathbf{b}^T\mathbf{p} \\
 & \text{s.t.} & \mathbf{p}^T\mathbf{A} \leq \mathbf{c}^T
\end{array}
$$

$\mathbf{A} \in \Re^{m\times n}$, where $n >> m$. Typical iteration:

1. Start with $I \subset \{1, \dots, n\}$
2. Solve the *Restricted/Relaxed Master Problem*:

$$
\begin{array}{lll}
(\text{ResMP}) & \min & \sum_{j\in I} c_j x_j \\
 & \text{s.t.} & \sum_{j\in I} \mathbf{A}_j x_j = \mathbf{b} \\
 & & x_j \geq \mathbf{0},\ j \in I
\end{array}
\qquad
\begin{array}{lll}
(\text{RelMP}) & \max & \mathbf{b}^T\mathbf{p} \\
 & \text{s.t.} & \mathbf{p}^T\mathbf{A}_j \leq \mathbf{c}_j^T\ j \in I
\end{array}
$$

   Let $(\mathbf{x}_I, \mathbf{p})$ be respective optimal solutions.
3. Check if $c_j - \mathbf{p}^T\mathbf{A}_j < 0$ for some $j \in \{1, \dots, n\}$.
4. If not, terminate: $(\mathbf{x}, \mathbf{p})$, where $x_j = 0$ for $j \notin I$, are optimal for (P)/(D)
5. Otherwise
   - add one (or more) indices j (with at least one $c_j - \mathbf{p}^T\mathbf{A}_j < 0$) to I;
   - if desired, remove one or more non-basic indices from I;
   - return to step 1.

The pricing problem

- If n is large, enumerating all columns of $\mathbf{A}$ in step 3 is impossible
- Sometimes, structure of the columns of $\mathbf{A}$ is such that we can easily solve the *pricing subproblem*:

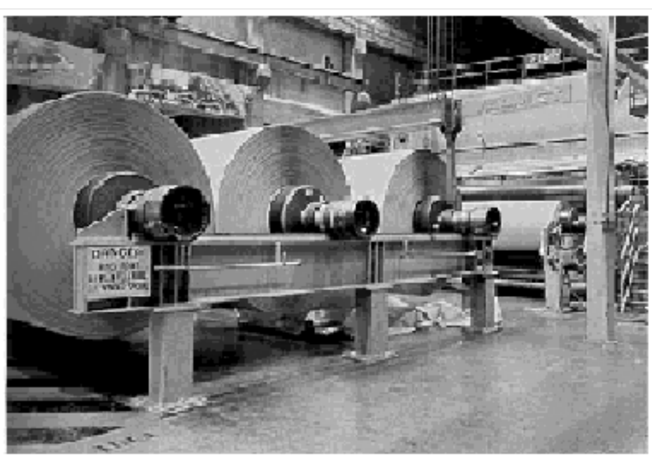
$$Z^{\star}(\mathbf{p}) = \min_{j \in \{1,\ldots,n\}} (c_j - \mathbf{p}^T \mathbf{A}_j)$$

  without explicitly examining each column
  - Note: if $Z^{\star}(\mathbf{p}) \geq 0$, then the algorithm terminates; o/w solution to the pricing problem gives us an entering variable
- The nature of the pricing problem depends on the setting; studied via examples
- The Cutting Stock Problem is one of the most famous, and simple, examples

Cutting stock problem

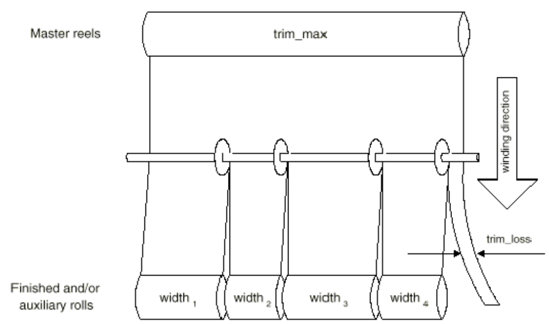
Introduction

- A paper company has a supply of large rolls of paper, each of width W
- Customer orders are for o_i rolls of width w_i, $i = 1, \ldots, m$ ($w_i \leq W$)



Cutting stock problem
Introduction

- The demand can be met by slicing a large roll in a certain way, called a **pattern**.
- For example, a large roll of width 100 can be cut into
  - 4 rolls each of width 25, or
  - 2 rolls each of width 35, with a waste of 30
  - etc.
- Goal: meet the demand with the lowest amount of waste



Solution approach I: L.V. Kantorovich
Formulation

1939 Russian, 1960 English
L.V. Kantorovich, "Mathematical Methods of Planning and Organising Production" *Management Science*, **6**, 366-422.

- $\mathcal{K}$: Set of available rolls.
- y^k: 1 if roll k is used (cut), 0 otherwise, $k \in \mathcal{K}$.
- x_i^k: number of times item i is cut on roll k, $k \in \mathcal{K}$, $i = 1, \ldots, m$.

Model:

$$
\begin{array}{lll}
\min & \sum_{k\in\mathcal{K}} y^k & \\
\text{s.t.} & \sum_{k\in\mathcal{K}} x_i^k \geq o_i & \text{for each item } i = 1, \ldots, m, \\
& \sum_i w_i x_i^k \leq W y^k & \text{for each roll } k \in \mathcal{K}, \\
& x_i^k \geq 0, \text{ integer} & k \in \mathcal{K}, \; i = 1, \ldots, m \\
& y^k \text{ binary} & k \in \mathcal{K}
\end{array}
$$

Solution approach I

AMPL model

The AMPL model for the problem:

```
# ----------------------------------------
# CUTTING STOCK USING KANTOROVICH'es MODEL
# ----------------------------------------

param roll_width > 0;         # width of raw rolls
param roll_number:=100;          # upper bound on the number of raw rolls
to be cut

set WIDTHS;                   # set of widths to be cut
param orders {WIDTHS} > 0;    # number of each width to be cut

var y{1..roll_number} binary;
var x{WIDTHS,1..roll_number}>=0, integer;

minimize Number: sum{k in 1..roll_number} y[k];

subj to Feas_Width{k in 1..roll_number}: sum {i in WIDTHS} i * x[i,k] <=
roll_width*y[k];
subj to Demand{i in WIDTHS}: sum{k in 1..roll_number}x[i,k]>=orders[i];

option cplex_options 'mipdisplay=2 mipinterval=10000 timing=1';
```


Solution approach I

Solution times[8]

Experiment I:

- o_i : uniform, between 1 and 100 (rand(100)+1);
- w_i : uniform, between 1 and 30 (rand(30)+1);
- Width of Roll, $W = 3000$;

Rolls	Items	constr	variables	CPU (s)
30	60	90	1830	1.99
50	100	150	5050	3.94
100	200	300	20100	24.50
200	400	600	80200	359.48

[8] Warning: here and beyond, results of 5-year old runs

Solution approach I

Solution times

Experiment II: Change width of the roll from 3000 to 150.

Rolls	Items	constr	variables	CPU (s)
70	10	80	770	3.62
140	20	160	2940	17.86
210	30	240	6510	40.03
280	40	320	11480	out of memory

The performance has deteriorated. Why?

How good is the LP relaxation?

Observation: $z_{LP} = \frac{\sum_i w_i o_i}{W}$.

The optimal solution of LP relaxation will satisfy:

- Choose y^k as small as possible. Therefore, for all k,

$$\sum_i w_i x_i^k = W y^k$$

- Choose x_i^k as small as possible. Therefore, for all i

$$\sum_K x_i^k = o_i$$

Objective value:

$$\sum_{k \in \mathcal{K}} y^k = \sum_{k \in \mathcal{K}} \frac{\sum_i w_i x_i^k}{W} = \sum_i \sum_{k \in \mathcal{K}} \frac{w_i}{W} x_i^k = \sum_i \frac{w_i}{W} \sum_{k \in \mathcal{K}} x_i^k = \sum_i \frac{w_i o_i}{W}$$

— relaxation might be good if W is very large compared to w_i, but not if their values are comparable.

Solution approach I

A difficult instance

Another example: $W = 273$

Quantity Ordered	Order Width (inches)
233	18
310	91
122	21
157	136
120	51

- LP relaxation: solved in less than 0.1s. Objective value 228.7106.
- IP: found solution with objective value 230, then got stuck enumerating the branch-and-bound tree

Solution approach II: Gilmore and Gomory

Set covering formulation

P.C. Gilmore and R.E. Gomory, "A linear programming approach to the cutting-stock problem," *Oper. Res.*, **8** (1961), pp. 849-859.
Idea: Each way to cut the roll into items is a *pattern*

- Pattern j characterized by

  a_{ij} = number of times item i is cut in pattern j, $i = 1, \ldots, m$
- For example, a large roll of width 100 can be cut into
  - 4 rolls each of width $w_i = 25$ (pattern j, $a_{ij} = 4$, all others 0)
  - 2 rolls each of width $w_k = 35$ (pattern l, $a_{kl} = 2$, all others 0)
- x_j = number of times pattern j is used

Solution approach II

Example

Example: An instance: $W = 100$, $m = 3$.

w_i	*pattern* 1	2	3	4	5	6	o_i
25	4	2	2	1	0	0	**150**
35	0	1	0	2	1	0	**200**
45	0	0	1	0	1	2	**300**

Formulation:

minimize $\sum_{j=1}^{6} x_j$

x_1	x_2	x_3	x_4	x_5	x_6	RHS
$4x_1 +$	$2x_2 +$	$2x_3 +$	$1x_4 +$	$0x_5 +$	$0x_6 \geq$	**150**
$0x_1 +$	$1x_2 +$	$0x_3 +$	$2x_4 +$	$1x_5 +$	$0x_6 \geq$	**200**
$0x_1 +$	$0x_2 +$	$1x_3 +$	$0x_4 +$	$1x_5 +$	$2x_6 \geq$	**300**

$x_j \geq 0$, integer, $j = 1, \ldots, n$.

Solution approach II

Formulation and relaxation

Set-covering formulation:

$$\begin{array}{lll} \min & \sum_{j=1}^{n} x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \geq o_i, & i = 1, \ldots, m, \\ & x_j \in \mathbb{Z}^+, & j = 1, \ldots, n \end{array}$$

Linear relaxation (LP):

$$\begin{array}{lll} \min & \sum_{j=1}^{n} x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j - s_i = o_i, & i = 1, \ldots m, \\ & x_j \geq 0, & j = 1, \ldots, n \\ & s_i \geq 0, & i = 1, \ldots, m \end{array}$$

- We will focus on solving the LP via column generation
  - Each variable corresponds to a different **feasible pattern**
  - Large-scale problem: n may be huge!
  - However, m is just the number of different orders

Cutting stock problem

Pricing subproblem

- Let $I \subset \{1, \ldots, n\}$, and let $(\mathbf{x}, \mathbf{p})$ solve the corresponding (ResMP) and (RelMP)
  - I should include all slacks, and enough patterns to ensure feasibility of (ResMP)
- Pricing subproblem:

$$Z^{\star}(\mathbf{p}) = \min_{j \in \{1, \ldots, n\}} (c_j - \mathbf{p}^T \mathbf{A}_j) = \min_{\mathbf{z} \text{ is a feasible pattern}} (1 - \mathbf{p}^T \mathbf{z})$$

- Characterizing feasible patterns: all vectors $\mathbf{z} \in \mathbb{Z}_+^m$ satisfying

$$\sum_{i=1}^{m} w_i z_i \leq W$$

- So, the pricing problem can be formulated as the following IP:

$$\begin{aligned} Z^{\star}(\mathbf{p}) = \min \quad & 1 - \textstyle\sum_{i=1}^{m} p_i z_i \\ \text{s.t.} \quad & \textstyle\sum_i w_i z_i \leq W \\ & z_i \in \mathbb{Z}_+ \ \forall i \end{aligned}$$

Cutting stock problem

Solving the pricing subproblem — Knapsack problem

- The pricing problem for the Cutting Stock Problem is equivalent to the following **Knapsack problem**:

$$\begin{aligned} \max \quad & \textstyle\sum_{i=1}^{m} \pi_i z_i \\ \text{s.t.} \quad & \textstyle\sum_i w_i z_i \leq W \\ & z_i \in \mathbb{Z}_+ \ \forall i \end{aligned}$$

- Computational results on random instances using the MIP solver from CPLEX:

n	CPU (s)
1,000	0.22
10,000	1.04
100,000	75.52

- More specialized algorithm can be used to solve the Knapsack problem even more efficiently in practice.

Cutting stock problem

How good is the LP bound?

Round Up Conjecture:

$$v_{IP} \leq \lceil v_{LP} \rceil ?$$

Unfortunately, this is not true:

- $W = 273$

$w_1 = 18$	$o_1 = 233$
$w_2 = 91$	$o_2 = 310$
$w_3 = 21$	$o_3 = 122$
$w_4 = 136$	$o_4 = 157$
$w_5 = 51$	$o_5 = 120$

- $v_{LP}(CG) = 228.9982$, $v_{IP} = 230$.

Modified Round Up Conjecture:

$$v_{IP} \leq \lceil v_{LP} \rceil + 1?$$

- This conjecture has not been answered.

Cutting stock problem

Getting integer solutions: Round-up heuristic

- Let x_j be the LP solution obtained from the column generation method — possibly fractional.
- Let $x'_j = \lceil x_j \rceil$ — integer.
- $\sum_j a_{i,j} x_j \geq o_i$ implies $\sum_j a_{i,j} x'_j \geq o_i$. So x'_j defined in this way is a feasible integer solution.
- How good is this heuristic?

$W = 273$

$w_1 = 18$	$o_1 = 233$
$w_2 = 91$	$o_2 = 310$
$w_3 = 21$	$o_3 = 122$
$w_4 = 136$	$o_4 = 157$
$w_5 = 51$	$o_5 = 120$

- $v_{LP}(CG) = 228.9982$; Round-Up produces a value of 231

Cutting stock problem

Round-up heuristic

	Round-Up	Fractional	**18**	**91**	**21**	**136**	**51**
cut	0	0.0000	15	0	0	0	0
cut	104	103.3333	0	3	0	0	0
cut	9	8.2363	0	0	13	0	0
cut	79	78.5000	0	0	0	2	0
cut	0	0.0000	0	0	0	0	5
cut	24	24.0000	1	0	0	0	5
cut	15	14.9286	14	0	1	0	0

Can you give a bound on $\sum_j x_j^I - \sum_j x_j^\star$?

Cutting stock problem

Exact solution approach

- For an exact solution, can apply B&B algorithm to the cutting stock problem
- Every subproblem is an instance of the cutting stock problem with lower/upper bounds on some of the variables
- LP relaxation of every subproblem can be solved by this column generation method

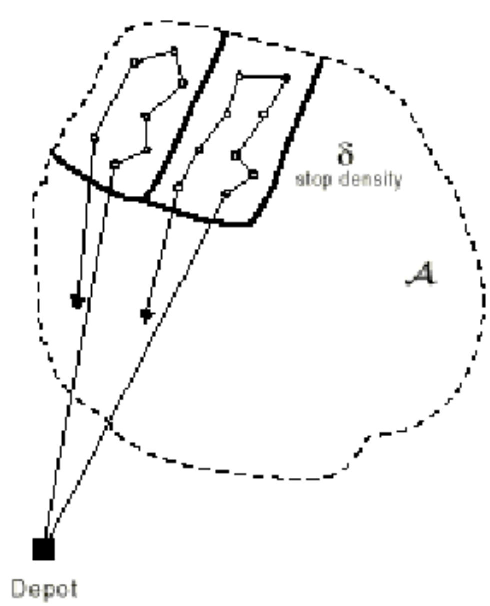
Other applications of this column generation technique
Vehicle routing

Vehicle Routing with time window or other types of constraint:

- A set of m customers.
- Each customer must be served within certain time window.
- Problem: Find a set of routes to serve all the customers, so that each customer will be visited by a vehicle within the stipulated time window.
- Likely objectives could be:
  - Minimize the total number of vehicles
  - Minimize the total cost (fixed per-vehicle cost, plus travel costs)

Note the "set covering" nature of this problem.

Other applications of this column generation technique
Vehicle routing



Other applications of this column generation technique

Vehicle routing "formulation"

- Variables: $x_j \in \{0, 1\}$ — binary, representing whether the jth route is taken by one of the trucks
  - Each variable/column represents a feasible route, i.e., a subset of customers, and the order in which they are visited, so that each visit occurs within appropriate time windows
- Parameters: $a_{ij} = 1$ if customer i is included in route j; 0 otherwise
- Formulation ("minimize number of vehicles" version):

$$\begin{array}{ll} \min & \sum_j x_j \\ \text{s.t.} & \sum_j a_{ij} x_j \geq 1, \;\; i = 1, \ldots, m \\ & x_j \in \{0, 1\} \; \forall j \end{array}$$

Other applications of this column generation technique

Vehicle routing — Column generation and pricing subproblem

- LP relaxation is hard to solve due to enormous number of variables — use column generation
- Solve LP relaxation with a small subset of columns/variables
- Obtain dual variables p_i for each demand point (can be interpreted as marginal profits of the customers)
- Reduced cost

$$\bar{c}_j = 1 - \sum_i a_{ij} p_i = 1 - \sum_{i:\text{route } j \text{ visits } i} p_i$$

- Pricing subproblem: find a route that:
  - satisfies feasibility constraints (meets the time window constraints);
  - total profits accrued by serving the demand points on the route is maximized

— a variant of the Traveling Salesman problem.