Column Generation Basics

Ed Klotz, Ph.D., Mathematical Optimization Specialist, Gurobi Optimization, New Trier East class of 1978

Greg Glockner, Ph.D., Technical Fellow, Gurobi Optimization, New Trier East class of 1988

May, 2024

GUROBI OPTIMIZATION

Question 1

Consider the LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & A_1 x = b \\ & x \geq 0 \end{aligned}$$

Suppose A_1 has 10000 constraints and 50000 variables. At any simplex method iteration, how many variables are basic?

- < 10000
- 10000
- More than 10000 but less than 50000
- 50000

Question 2

Consider the LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & A_2 x = b \\ & x \geq 0 \end{aligned}$$

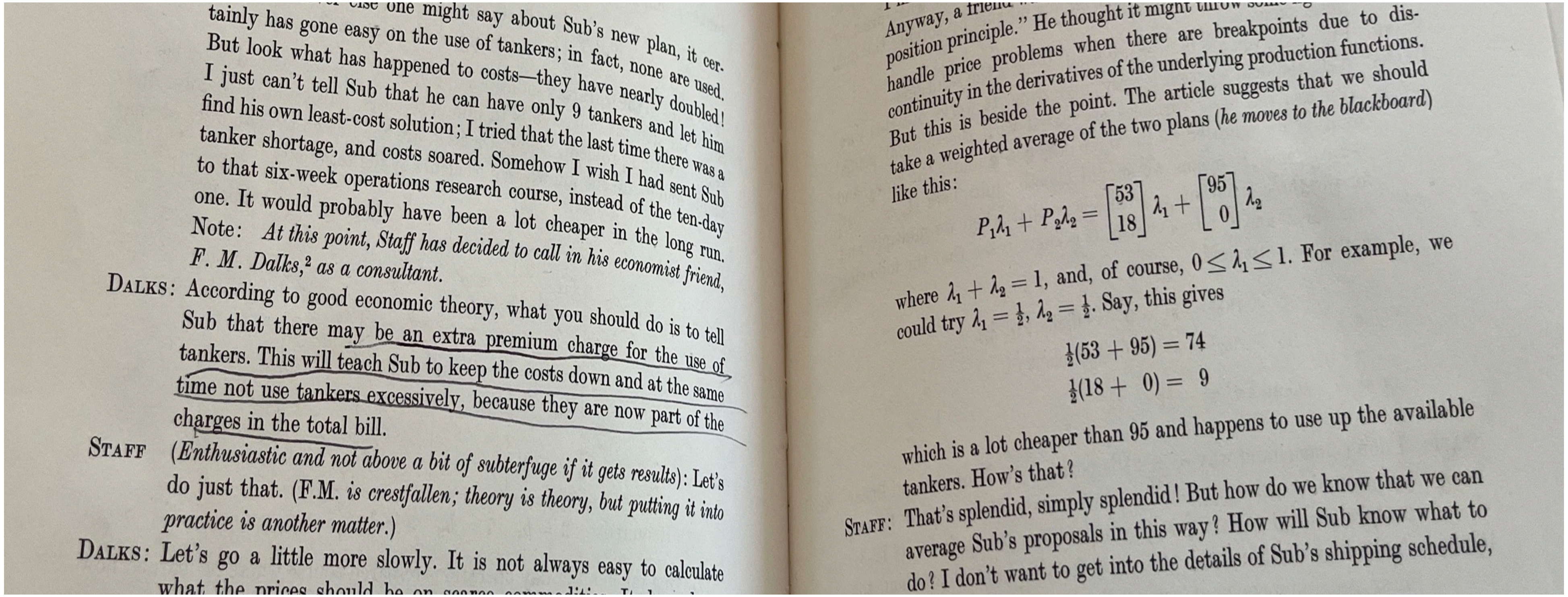
Suppose A_2 has 10000 constraints and 10^{30} variables. At any simplex method iteration, how many variables are basic?

- < 10000
- 10000
- More than 10000 but less than 10^{30}
- 10^{30}

Column Generation, what’s that all about?

It didn’t make it on Broadway, but…

- From Dantzig, Linear Programming and Extensions, 1963



else one might say about Sub’s new plan, it certainly has gone easy on the use of tankers; in fact, none are used. But look what has happened to costs—they have nearly doubled! I just can’t tell Sub that he can have only 9 tankers and let him find his own least-cost solution; I tried that the last time there was a tanker shortage, and costs soared. Somehow I wish I had sent Sub to that six-week operations research course, instead of the ten-day one. It would probably have been a lot cheaper in the long run.

Note: *At this point, Staff has decided to call in his economist friend, F. M. Dalks,*[2] *as a consultant.*

DALKS: According to good economic theory, what you should do is to tell Sub that there may be an extra premium charge for the use of tankers. This will teach Sub to keep the costs down and at the same time not use tankers excessively, because they are now part of the charges in the total bill.

STAFF (*Enthusiastic and not above a bit of subterfuge if it gets results*): Let’s do just that. (*F.M. is crestfallen; theory is theory, but putting it into practice is another matter.*)

DALKS: Let’s go a little more slowly. It is not always easy to calculate what the prices should be on scarce commodities. It

Anyway, a friend

position principle.” He thought it might throw

handle price problems when there are breakpoints due to discontinuity in the derivatives of the underlying production functions. But this is beside the point. The article suggests that we should take a weighted average of the two plans (*he moves to the blackboard*) like this:

$$P_1\lambda_1 + P_2\lambda_2 = \begin{bmatrix} 53 \\ 18 \end{bmatrix} \lambda_1 + \begin{bmatrix} 95 \\ 0 \end{bmatrix} \lambda_2$$

where $\lambda_1 + \lambda_2 = 1$, and, of course, $0 \leq \lambda_1 \leq 1$. For example, we could try $\lambda_1 = \frac{1}{2}$, $\lambda_2 = \frac{1}{2}$. Say, this gives

$$\tfrac{1}{2}(53 + 95) = 74$$
$$\tfrac{1}{2}(18 + \ \ 0) = \ \ 9$$

which is a lot cheaper than 95 and happens to use up the available tankers. How’s that?

STAFF: That’s splendid, simply splendid! But how do we know that we can average Sub’s proposals in this way? How will Sub know what to do? I don’t want to get into the details of Sub’s shipping schedule,

Act 1

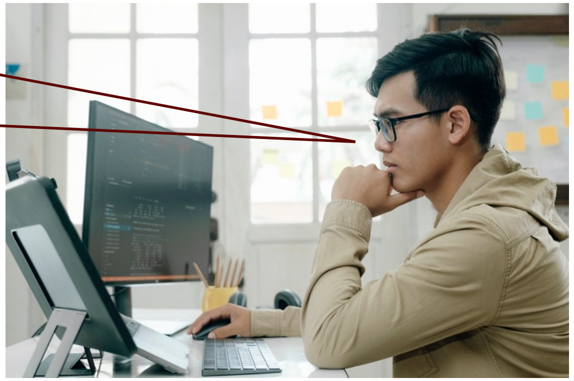
We need to expand into a new line of business.

All of our high tech business lines are losing money. We need to move into something more low tech and boring that is quietly profitable

Act 1

I’ve decided we are going to expand into clothing, lumber, steel and paper production.   In order to do that, we’ll need to be able to solve cutting stock problems.

What is the cutting stock problem?



Act 1

The one dimensional cutting stock problem involves cutting rectangles of smaller widths out of generic rectangles (the “stock”) with the same standard width. Here’s an example.

Stock rolls

100

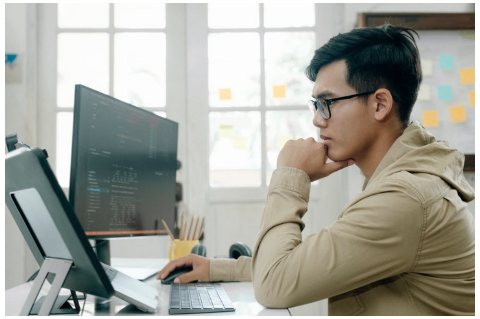
100

100

.
.
.

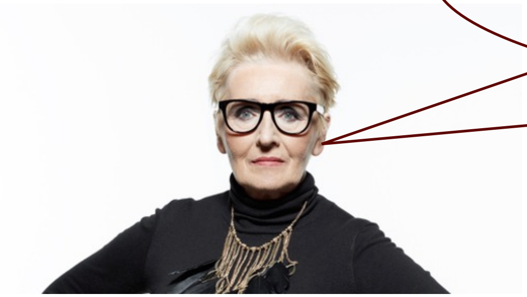
100

Product rolls	Demand
17	211
31	395
38	610
45	97

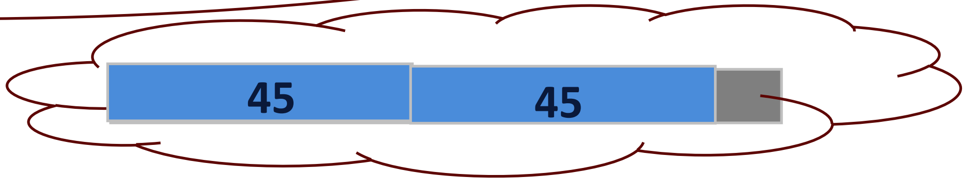


Modified problem from Linear Programming by Chvatal

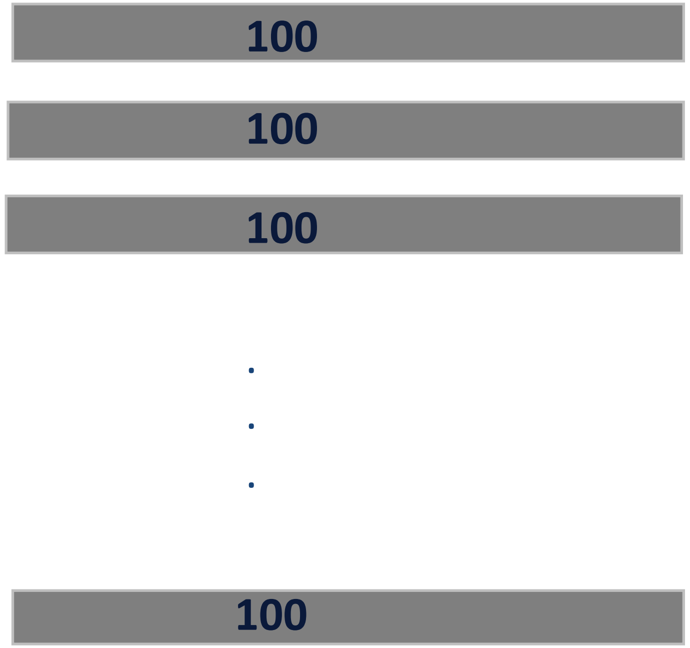
Act 1



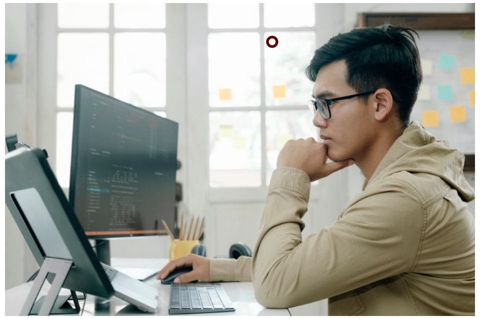
The costs of the various cut patterns is the same. You mission, and you have to accept it, is to choose a collection of cut patterns that uses as few stock rolls as possible while meeting demand.



Stock rolls



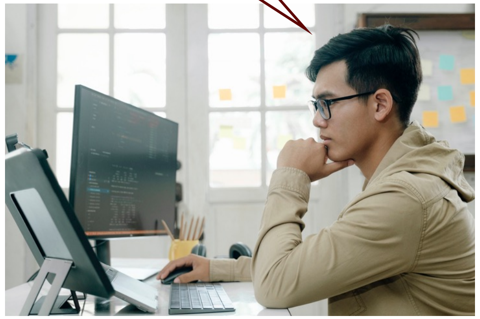
Product rolls	Demand
17	211
31	395
38	610
45	97



Act 1

Hmm. 529 sounds expensive. And look at all that wasted material. I bet less expensive configurations are possible. Try harder.

Here’s a solution that uses 529 stock rolls



Cut pattern	Quantity	Demand
17 17 17 17 17	43	211
31 31 31	132	395
38 38	305	610
45 45	49	97
100	529	

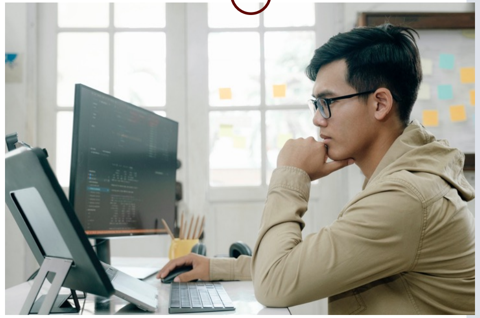
Act 1

38 | 38 | 17

She complained about too much waste. The first and third patterns are the most wasteful. Aha! Here's a single pattern that uses rolls of length 17 but has much less waste.

I can use this less wasteful pattern to reduce the usage of the most wasteful patterns. And voila, stock rolls used drops from 529 to 486. That should satisfy him.

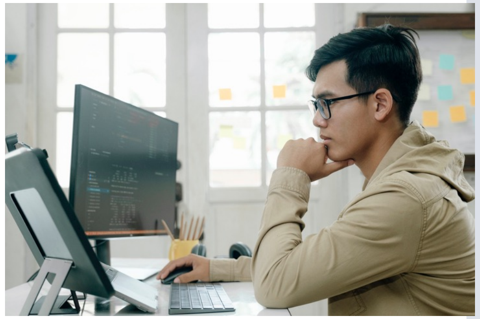
Cut pattern	Quantity	Demand
17 17 17 17 17	~~43~~ 0	211
31 31 31	132	395
38 38	~~305~~ 94	610
45 45	49	97
38 38 17	211	
100	486	



Act 1

Accounting says our profit margins will still be small with that manufacturing cost. And we're still using that third pattern 94 times despite all the waste.

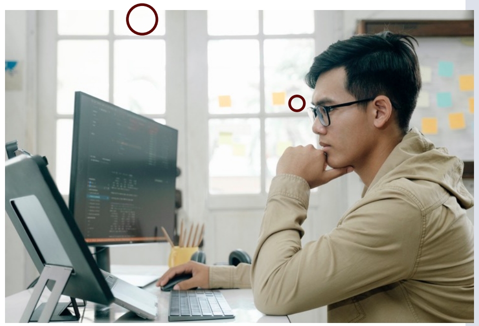
Cut pattern	Quantity	Demand
17 17 17 17 17	0	211
31 31 31	132	395
38 38	94	610
45 45	49	97
38 38 17	211	
100	486	



Act 1

This is getting complicated.
What else can I do?

Cut pattern	Quantity	Demand
17 17 17 17 17	0	211
31 31 31	132	395
38 38	94	610
45 45	49	97
38 38 17	211	
100	486	

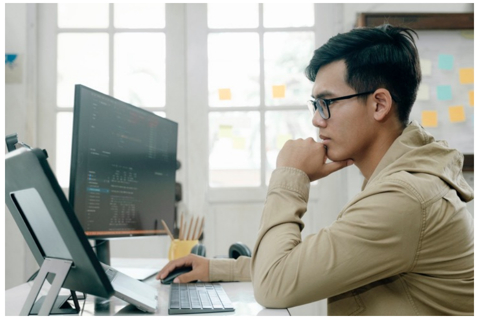


Act 1

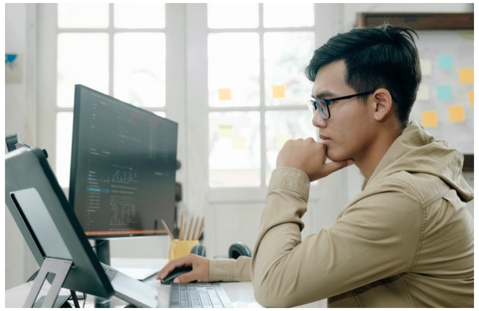
< 2 days later>

I still haven't found a cheaper solution. Help me somebody!!

Cut pattern	Quantity	Demand
17 17 17 17 17	0	211
31 31 31	132	395
38 38	94	610
45 45	49	97
38 38 17	211	
100	486	



Act 2



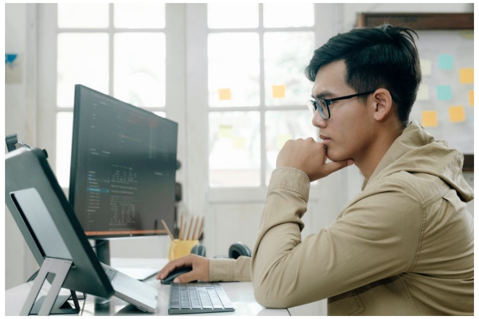
Excuse me, but I couldn't help but notice that you have been working on cutting stock problems lately.

Cut pattern	Quantity	Demand
17 17 17 17 17	0	211
31 31 31	132	395
38 38	94	610
45 45	49	97
38 38 17	211	
100	486	

Act 2

Think about what you did to reduce the cost with pattern 5. While you viewed it as reducing waste, you could also view it as reducing cost. You added 211 rolls cut with pattern 5, but you eliminated 43 rolls cut with pattern 1 and 211 rolls with pattern 3, for a savings of 43 = 529 - 486

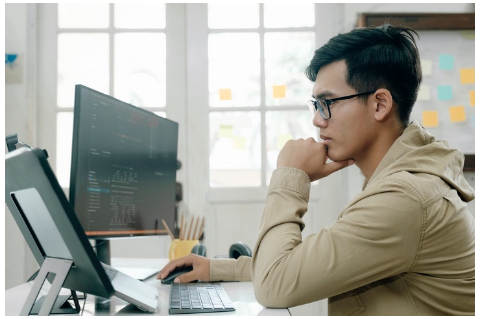
#	Cut pattern	Quantity	Change	Demand
1	17 17 17 17 17	~~43~~ $\boxed{0}$	-43	211
2	31 31 31	132		395
3	38 38	~~305~~ $\boxed{94}$	-211	610
4	45 45	49		97
5	38 38 17	211	211	
	100	486	-43	



Act 2

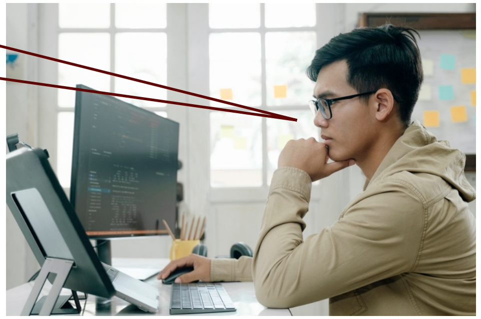
That worked, but it then was harder to find another pattern that yielded more savings.   What you need to do is use an algorithm to efficiently search for additional cut patterns that yield more savings.   That's where column generation can help you. It will find new patterns where the reduction in stock rolls it enables exceeds the increased use of the new pattern.

#	Cut pattern	Quantity	Change	Demand
1	17 17 17 17 17	0	-43	211
2	31 31 31	132		395
3	38 38	94	-211	610
4	45 45	49		97
5	38 38 17	211	211	
	100	486	-43	



Act 2

So how do you know about column generation?



It’s taught in the second week of onboarding at our waste management company.   We need to figure out the most cost efficient route to meet our customer pickups. Column generation is very helpful for vehicle routing problems, either pickup like we do or delivery like all sorts of companies do.

Intermission

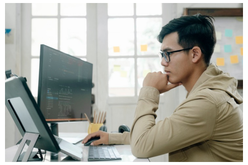
What are the takeaways from Acts 1 and 2?

- There was no mention of mathematical programming, linear algebra, or other optimization terminology
- An iterative procedure emerged involving a dialog between the Boss and the Software Engineer. Each person passed information to the other that helped refine the best solution
- Even a very small cutting stock problem is not that easy to solve.

Act 3: Column Generation and the Cutting Stock Problem

Standard form LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \qquad A = (A_B, A_N) \\ & x \geq 0 \end{aligned}$$



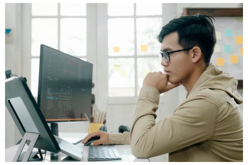
The reduced cost calculation can be viewed in one of two ways.

$$\bar{c}_j = c_j - c_B^T A_B^{-1} A_j = \boxed{c_j - \overbrace{(c_B^T A_B^{-1})}^{y^T} A_j} = \boxed{c_j - c_B^T \overbrace{(A_B^{-1} A_j)}^{\bar{A}_j}}$$

The most common one is $z_j = c_j - y^T A_j$ where y are the dual variables that solve the linear system $y^T A_B = c_B^T$, and hence have values $y^T = c_B^T A_B^{-1}$

The second one is $z_j = c_j - c_B^T \bar{A}_j$ where $\bar{A}_j$ is the representation of A_j relative to the current basis A_B. Let's have a closer look at this one

Act 3: Column Generation and the Cutting Stock Problem



This is essentially what you did when you figured out how to reduce the cost from 529 to 486

Now you need to figure out how to automate that the process you did manually with column generation.

Standard form LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

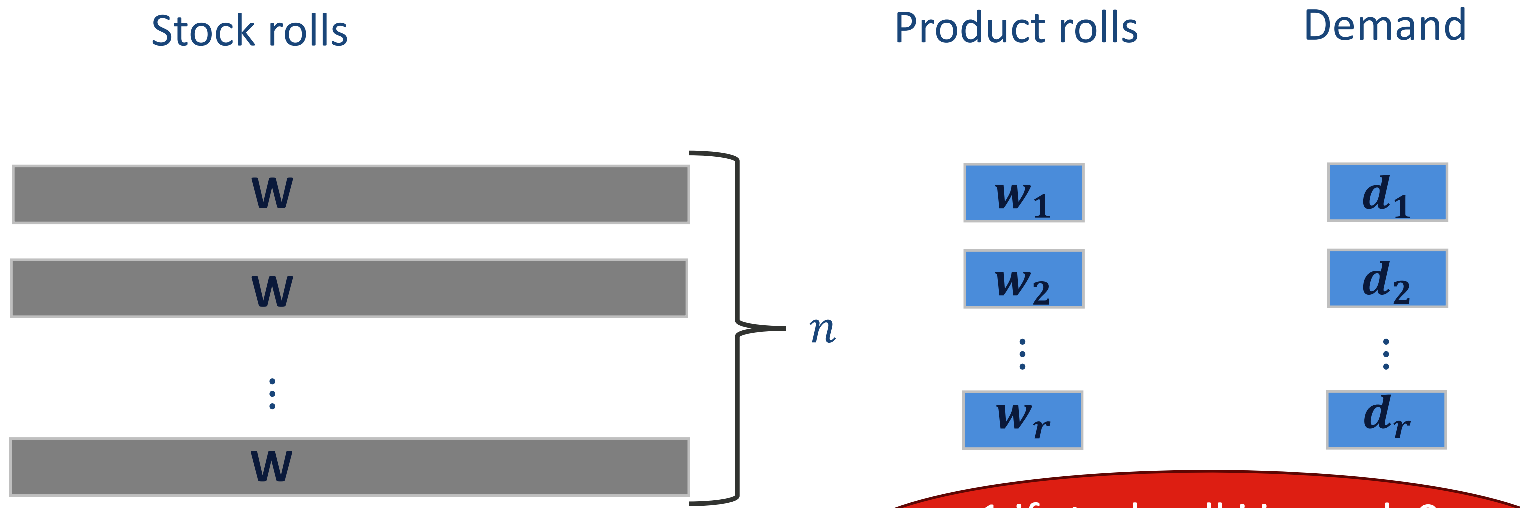
Cost of increasing variable j

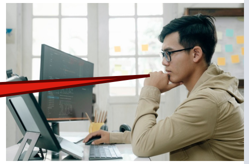
Adjustment to cost to reflect changes in basic variables

$$z_j = c_j - c_B^T \bar{A}_j$$

Cut pattern	Quantity		Change	Demand
17 17 17 17 17	$\boxed{0}$	$A_B^{-1}A_j$	-43	211
31 31 31	132			395
38 38	$\boxed{94}$	$A_B^{-1}A_j$	-211	610
45 45	49			97
38 38 17	211	c_j	211	
100	486		-43	

Act 3: Column Generation and the Cutting Stock Problem





What is n?

A compact formulation:

1 if stock roll j is used, 0 otherwise

# of product roll i cut from stock roll j

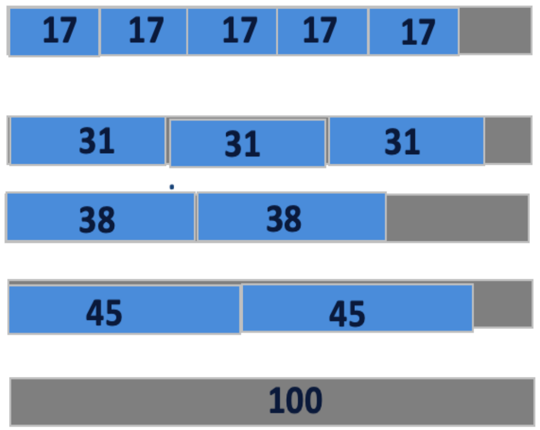
$$\begin{array}{lll} \min & \sum_{j=1}^{n} y_j & \\ s.t. & \sum_{i=1}^{r} w_i\, x_{ij} \leq W y_j & j = 1, \ldots, n \\ & \sum_{j=1}^{n} \; x_{ij} \geq d_i & i = 1, \ldots, r \\ & \; x_{ij} \geq 0, \text{integer}; y_j \text{ binary} & \end{array}$$

// Cannot exceed stock roll width

// Satisfy demand for each type of product roll

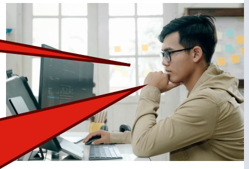
Act 3: Column Generation and the Cutting Stock Problem

$$\begin{aligned}
\min \quad & \sum_{j=1}^{n} y_j \\
s.t. \quad & \sum_{i=1}^{r} w_i\, x_{ij} \le W y_j \qquad j = 1, \dots, n \\
& \sum_{j=1}^{n} x_{ij} \ge d_i \qquad\qquad i = 1, \dots, r \\
& x_{ij} \ge 0, \text{integer}; y_j \text{ binary}
\end{aligned}$$



Cut pattern	Quantity	Demand
17 17 17 17 17	43	211
31 31 31	132	395
38 38	305	610
45 45	49	97
100	529	

What is n?



Ah, I see. I just need to look for solutions better than that obvious one that use fewer stock rolls. So n = $\sum_{i=1}^{r} ceil(d_i/floor(W/w_i))$

Think about that simple first solution you proposed. You just cut a single type of product roll from each stock roll until you met demand.

Act 3: Column Generation and the Cutting Stock Problem

$$
\begin{aligned}
\min \quad & \sum_{j=1}^{n} y_j \\
s.t. \quad & \sum_{i=1}^{r} w_i \, x_{ij} \le W y_j \quad && j = 1, \dots, n \\
& \sum_{j=1}^{n} x_{ij} \ge d_i && i = 1, \dots, r \\
& x_{ij} \ge 0, \text{integer}; \, y_j \text{ binary}
\end{aligned}
$$

We have this compact formulation. Why do we need column generation?

What do you mean weakness of the formulation?

This formulation will work fine for small cutting stock problems like the one we examined. But for larger problems with more different types of product rolls, solving this version of the model can become problematic due to the inherent weakness of the formulation.

Weakness of a MIP formulation

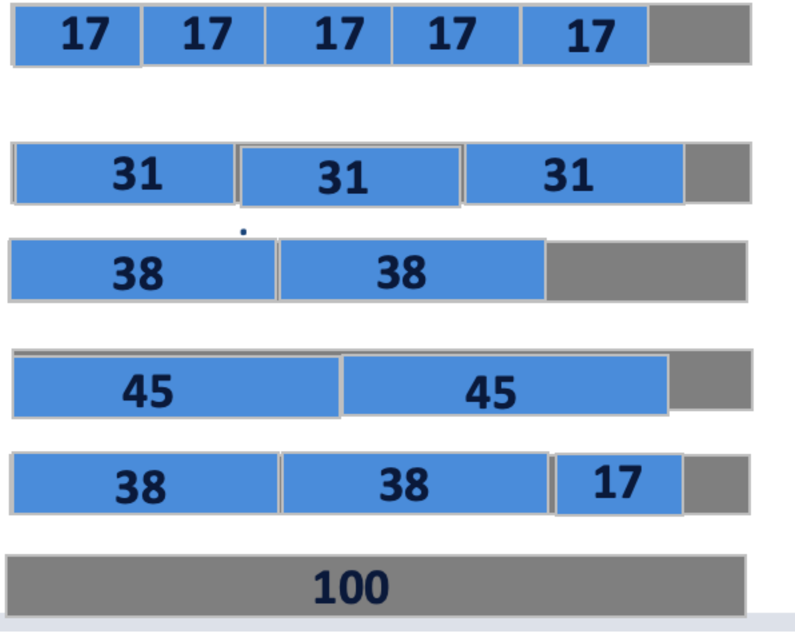
- Discussed in previous tech talk “Converting Weak to Strong MIP Formulations”, parts I and II
  - https://www.gurobi.com/events/tech-talk-chat-converting-weak-to-strong-mip-formulations/
  - https://www.gurobi.com/events/tech-talk-chat-converting-weak-to-strong-mip-formulations-part-ii/
  - Upcoming book chapter: Klotz, E. and Oberdieck, R. (2024 forthcoming). Converting Weak to Strong MIP Formulations: A Practitioner’s Guide. In: Hamid, F. (ed.) Optimization Essentials: Theory, Tools, and Applications. International Series in Operations Research & Management Science, vol 353. Springer, Singapore. https://doi.org/10.1007/978-981-99-5491-9_4
- Consider the level of disconnect between the physical systems modeled by the MIP formulation and its LP relaxation

Act 3: Column Generation and the Cutting Stock Problem

$$
\begin{array}{lll}
\min & \sum_{j=1}^{n} y_j & \\
s.t. & \sum_{i=1}^{r} w_i\, x_{ij} \le W y_j & j = 1, \dots, n \\
 & \sum_{j=1}^{n} \; x_{ij} \ge d_i & i = 1, \dots, r
\end{array}
$$

$x_{ij} \;\ge 0$, integer; y_j binary

Cut pattern



What do you mean weakness of the formulation?

We’ve seen how the system associated with the MIP formulation works. Individual cut patterns can result in wasted material

That no longer holds in the LP relaxation. We can reassemble wasted material into product rolls at no cost. Let’s see how that works

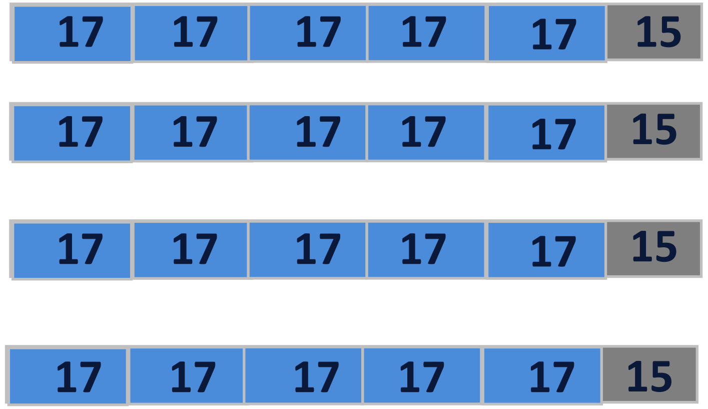
Act 3: Column Generation and the Cutting Stock Problem

$$\begin{aligned} \min \quad & \textstyle\sum_{j=1}^{n} y_j \\ s.t. \quad & \textstyle\sum_{i=1}^{r} w_i\, x_{ij} \le W y_j \quad && j = 1, \dots, n \\ & \textstyle\sum_{j=1}^{n} \; x_{ij} \ge d_i && i = 1, \dots, r \end{aligned}$$

$$x_{ij} \;\ge 0, \text{integer}; y_j \text{ binary}$$

Ah, so the fractional rolls still count towards meeting demand in the LP relaxation model.

Algebraically, consider cutting only product roll i=1, namely rolls of length 17; this results in waste of length 15 in each associated stock roll. Consider 4 such identical rolls.



17 17 17 17 17 15

17 17 17 17 17 15

17 17 17 17 17 15

17 17 17 17 17 15

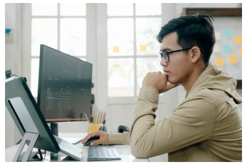
The LP relaxation solution will set x_{1j}=5 $^{15}/_{17}$ and y_j = 1 for j = 1, … 4, resulting in a total of $\sum_{j=1}^{4}\; x_{ij}$=23 product rolls and waste of 4*15 – 3*17 = 9

The MIP solution generates 20 product rolls of length 17, with a total waste of 4*15 =60

Act 3: Column Generation and the Cutting Stock Problem

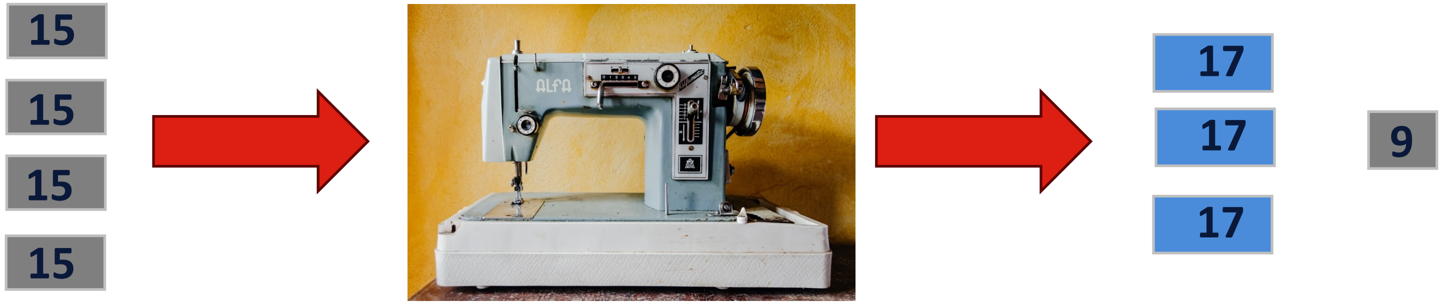
$$\begin{aligned}
\min \quad & \textstyle\sum_{j=1}^{n} y_j \\
s.t. \quad & \textstyle\sum_{i=1}^{r} w_i\, x_{ij} \le W y_j \quad & j = 1, \dots, n \\
& \textstyle\sum_{j=1}^{n} \; x_{ij} \ge d_i \quad & i = 1, \dots, r
\end{aligned}$$

$$x_{ij} \;\ge 0, \text{integer}; y_j \text{ binary}$$



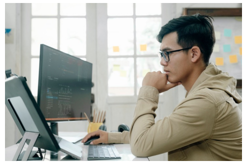
Visually, relaxing integrality introduces a new zero cost process that stitches together waste material shorter than any product roll into a legitimate product roll

17	17	17	17	17
17	17	17	17	17
17	17	17	17	17
17	17	17	17	17



GUROBI OPTIMIZATION

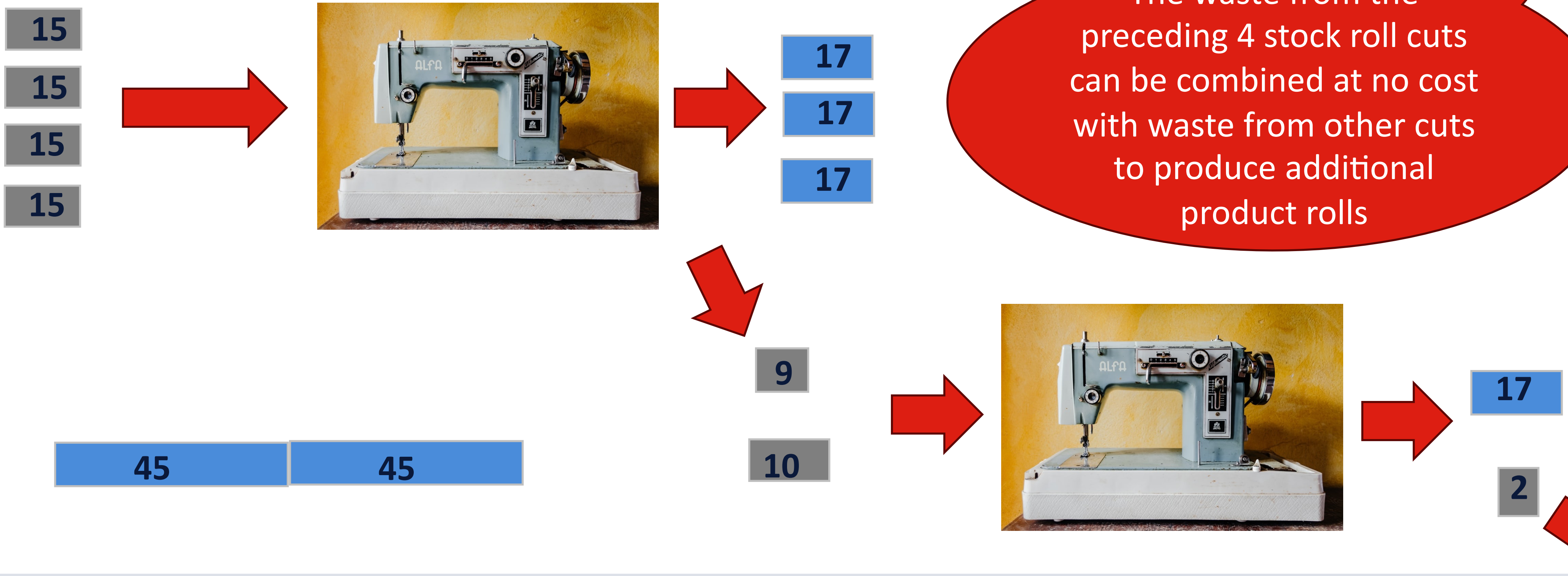
Act 3: Column Generation and the Cutting Stock Problem



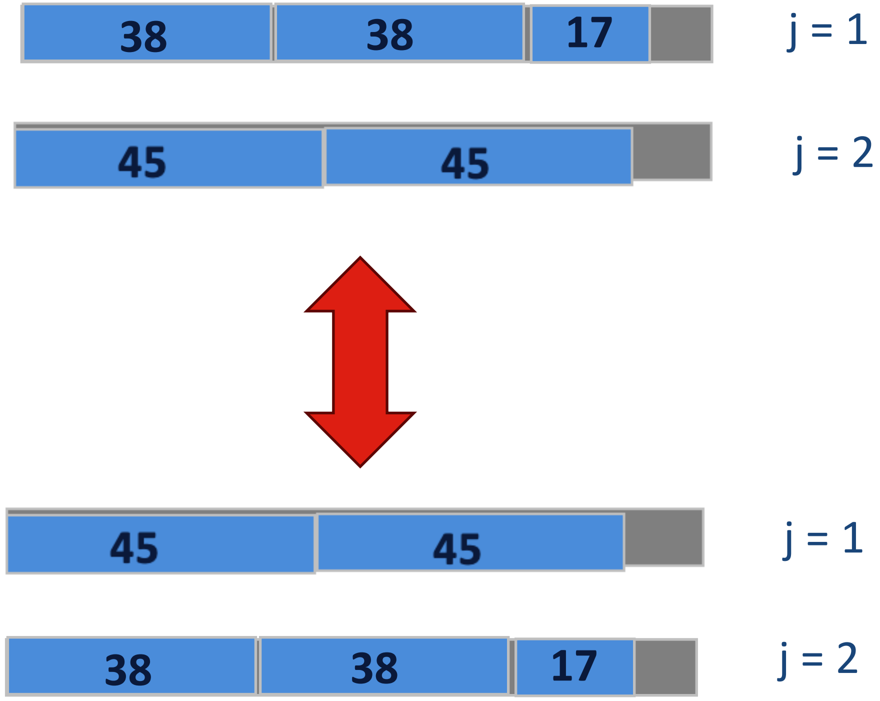
$$\begin{array}{lll} \min & \sum_{j=1}^{n} y_j & \\ s.t. & \sum_{i=1}^{r} w_i\, x_{ij} \le W y_j & j = 1, \dots, n \\ & \sum_{j=1}^{n} x_{ij} \ge d_i & i = 1, \dots, r \end{array}$$

$x_{ij} \ge 0$, integer; y_j binary

The waste from the preceding 4 stock roll cuts can be combined at no cost with waste from other cuts to produce additional product rolls

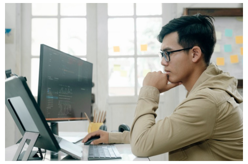


Act 3: Column Generation and the Cutting Stock Problem



$$
\begin{array}{lll}
\min & \sum_{j=1}^{n} y_j & \\
s.t. & \sum_{i=1}^{r} w_i \, x_{ij} \le W y_j & j = 1, \dots, n \\
 & \sum_{j=1}^{n} \; x_{ij} \ge d_i & i = 1, \dots, r \\
\end{array}
$$

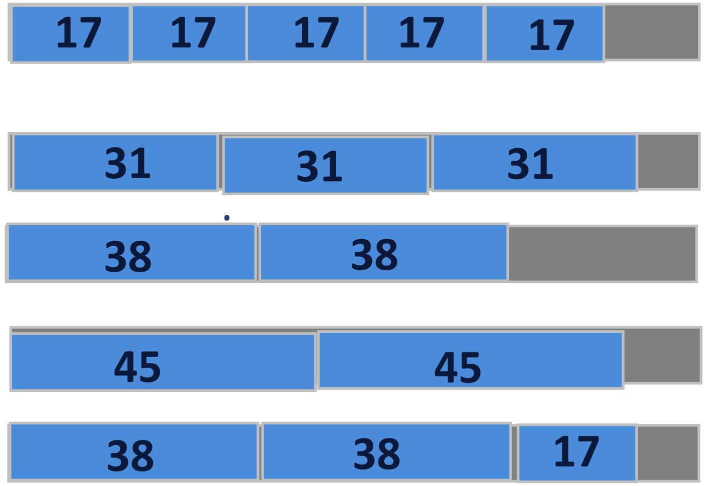
$$x_{ij} \ \ge 0, \text{integer}; \, y_j \text{ binary}$$



Symmetry is another source of weakness in this formulation. The indexing of the rolls is arbitrary and interchangeable

Act 3: Column Generation and the Cutting Stock Problem

I think I see how to formulate the model to use column generation. I'll implicitly consider all feasible cut patterns and encode the number of product rolls in each pattern. I can't explicitly enumerate all possible encodings, but I can enumerate enough to create a restricted master problem, then let the subproblem efficiently find other good encoded cut patterns



Cut pattern	Encoding
17 17 17 17 17	(5, 0, 0, 0)
31 31 31	(0, 3, 0, 0)
38 38	(0, 0, 2, 0)
45 45	(0, 0, 0, 2)
38 38 17	(1, 0, 2, 0)

Act 3: Column Generation and the Cutting Stock Problem

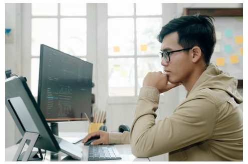
$$\begin{aligned} \min \quad & \textstyle\sum_{j\in K} z_j \\ & \textstyle\sum_{j\in K} p_{ij} z_j \ge d_i \\ & z_j \ge 0\,, \text{integer} \end{aligned}$$

(Implicit) set of all possible cut patterns

Number of product rolls i in encoded pattern j

Total demand for product rolls i

Number of stock rolls cut in pattern j



Here's the MIP with all possible cut patterns

Very good. Notice that each cut pattern is unique, so there's no symmetry like in the previous formulation. And relaxing integrality doesn't allow me to combine wasted material into legitimate product rolls.

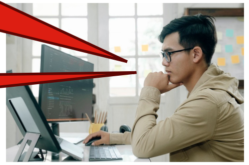
Act 3: Column Generation and the Cutting Stock Problem

Full master problem:

$$\begin{aligned}
\min \quad & \sum_{j\in J} z_j + \sum_{j\in K/J} z_j \\
y: \quad & \sum_{j\in J} p_{ij} z_j + \sum_{j\in K/J} p_{ij} z_j \ge d_i \\
& z_j \ge 0
\end{aligned}$$

~~, integer~~

Now we relax integrality and separate the patterns that go into the restricted master problem (RMP).

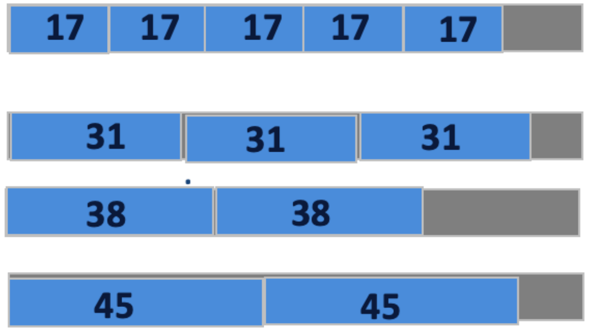


And a feasible basic solution is easily available for the RMP

Restricted master problem:

$$\begin{aligned}
\min \quad & \sum_{j\in J} z_j \\
y: \quad & \sum_{j\in J} p_{ij} z_j \ge d_i \\
& z_j \ge 0
\end{aligned}$$

Cut pattern



Encoding

(5, 0, 0, 0)

(0, 3, 0, 0)

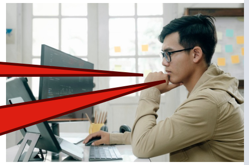
(0, 0, 2, 0)

(0, 0, 0, 2)

Act 3: Column Generation and the Cutting Stock Problem

Restricted master problem:

$$\begin{array}{lrl} \min & \sum_{j \in J} & z_j \\ y: & \sum_{j \in J} & p_{ij} z_j \geq d_i \\ & & z_j \geq 0 \end{array}$$



Now I just need to automate the thought process I used to find an improving cut pattern

I implicitly used the reduced cost computation $c_j - c_B^T(A_B^{-1}A_j)$, but for the subproblem, the equivalent computation $c_j - (c_B^T A_B^{-1})A_j$ works better

Subproblem:

$\min 1^T p$

$\max y^T p$

$\sum_{i=1}^{r} w_i \, p_i \leq W$

$p \geq 0$, integer

Cut pattern	Quantity	Change	Demand
17 17 17 17 17	$\cancel{43}$ $\boxed{0}$	-43 $A_B^{-1}A_j$	211
31 31 31	132		395
38 38	$\cancel{305}$ $\boxed{94}$	-211 $A_B^{-1}A_j$	610
45 45	49		97
38 38 17	211	211 c_j	
100	486	-43	

Act 3: Column Generation and the Cutting Stock Problem

Subproblem:

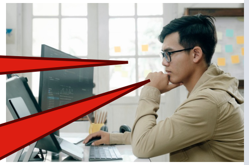
$$\max\ y^T p$$

$$\sum_{i=1}^{r} w_i\, p_i \le W$$

$$p \ \ \ge 0, \text{ integer}$$

$$p_q, q \in K/J$$

The subproblem is an integer program with one variable per product roll. It is a single constraint knapsack problem that’s easy to solve



Its optimal solution comes from the implicit cut patterns in the full master problem.   If that new pattern has a negative reduced cost, the objective of the restricted master problem will improve.

Restricted master problem:

$$\min \qquad \sum_{j\in J} z_j + \ z_q$$

$$y\!: \qquad \sum_{j\in J} p_{ij} z_j + p_{iq} z_q \ge d_i$$

$$z_j, z_q \ge 0$$

Full master problem:

$$\min \qquad \sum_{j\in J} z_j + \sum_{j\in K/J} z_j$$

$$y\!: \qquad \sum_{j\in J} p_{ij} z_j + \sum_{j\in K/J} p_{ij} z_j \ge d_i$$

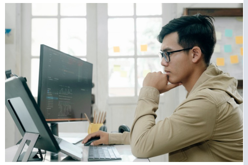
$z_j \ \ge 0$ ~~, integer~~

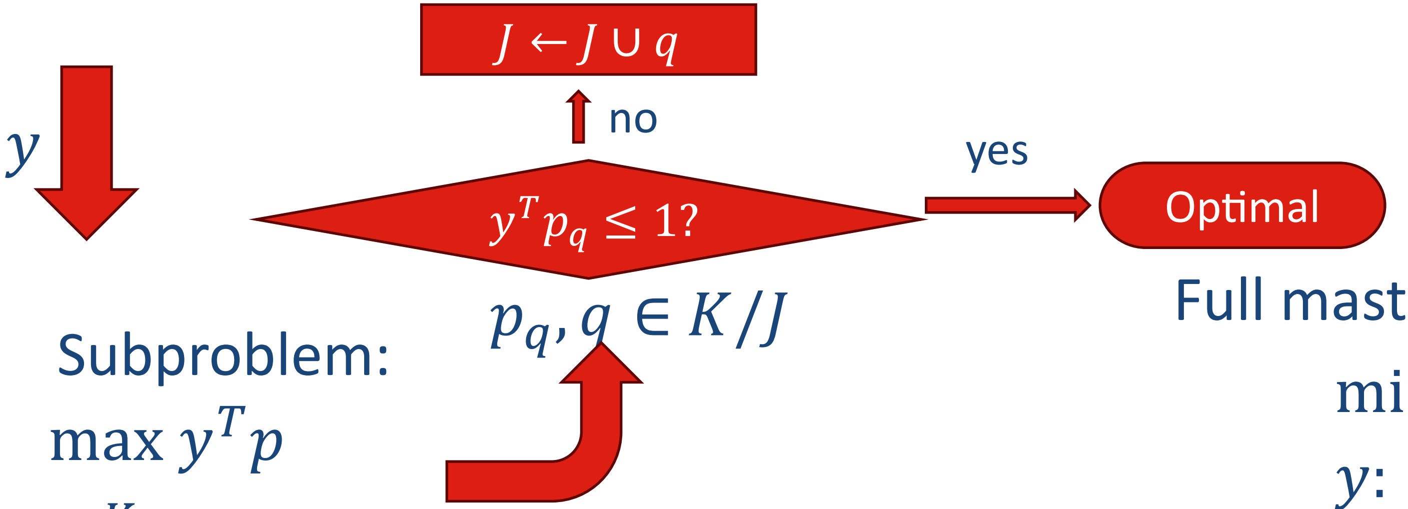
Act 3: Column Generation and the Cutting Stock Problem

Restricted master problem:

$$\begin{array}{ll} \min & \sum_{j\in J} z_j \\ y: & \sum_{j\in J} p_{ij} z_j \ge d_i \\ & z_j, \ge 0 \end{array}$$

I can repeat this process, and when the best cut pattern no longer has a favorable reduced cost, I have the optimal solution to both the restricted and full master problem.





Subproblem:

$\max y^T p$

$\sum_{i=1}^{K} w_i p_i \le W$

$p \ge 0$, integer

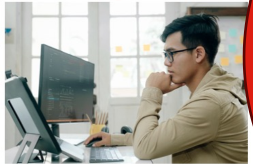
Full master problem:

$$\begin{array}{ll} \min & \sum_{j\in J} z_j + \sum_{j\in K/J} z_j \\ y: & \sum_{j\in J} p_{ij} z_j + \sum_{j\in K/J} p_{ij} z_j \ge d_i \\ & z_j \ge 0 \end{array}$$ ~~, integer~~

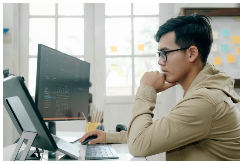
Act 3: Column Generation and the Cutting Stock Problem

I still haven't found a cheaper solution. Help me somebody!!

Cut pattern	Quantity	Demand
17 17 17 17 17	0	211
31 31 31	132	395
38 38	94	610
45 45	49	97
38 38 17	211	
100	486	



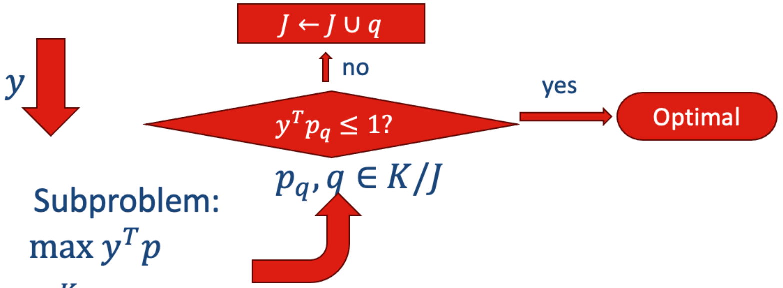
Now I see why you said I needed to automate the process of finding better solutions. The subproblem solve efficiently searches for the best cutting pattern among the potential huge number of implicit patterns in the master problem.



Restricted master problem:

$$\begin{aligned} \min \quad & \textstyle\sum_{j \in J} z_j \\ y: \quad & \textstyle\sum_{j \in J} p_{ij} z_j \geq d_i \\ & z_j, \geq 0 \end{aligned}$$

Exactly. There are too many patterns to store explicitly, and enumerative search would be hopelessly time consuming



Subproblem:

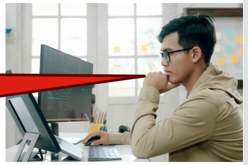
$$\begin{aligned} & \max y^T p \\ & \textstyle\sum_{i=1}^{K} w_i \, p_i \leq W \\ & p \;\; \geq 0, \text{integer} \end{aligned}$$

Full master problem:

$$\begin{aligned} \min \quad & \textstyle\sum_{j \in J} z_j + \sum_{j \in K/J} z_j \\ y: \quad & \textstyle\sum_{j \in J} p_{ij} z_j + \sum_{j \in K/J} p_{ij} z_j \geq d_i \\ & z_j \;\; \geq 0 \; \cancel{\text{, integer}} \end{aligned}$$

Act 3: Column Generation and the Cutting Stock Problem

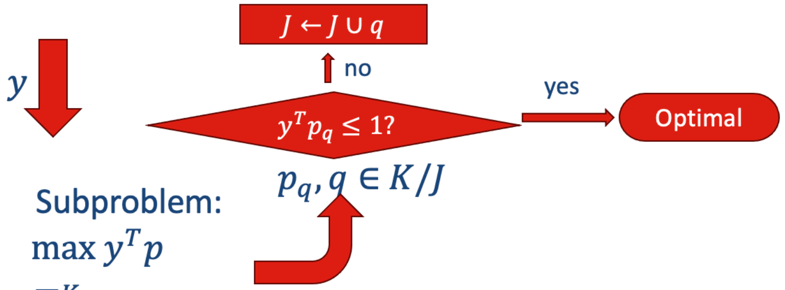
But wait a minute. Column generation has solved the full master problem LP. That might have fractional solutions, and I need an integer number for each cut pattern used.



Try solving it and see what you get.

Restricted master problem:

$$\begin{array}{ll} \min & \sum_{j\in J} \; z_j \\ y: & \sum_{j\in J} p_{ij} z_j \geq d_i \\ & \quad z_j, \geq 0 \end{array}$$



Subproblem:

$\max y^T p$

$\sum_{i=1}^{K} w_i \, p_i \leq W$

$p \;\; \geq 0$, integer

Full master problem:

$$\begin{array}{ll} \min & \sum_{j\in J} \; z_j + \sum_{j\in K/J} z_j \\ y: & \sum_{j\in J} p_{ij} z_j + \sum_{j\in K/J} p_{ij} z_j \geq d_i \\ & \quad z_j \;\; \geq 0 \end{array}$$

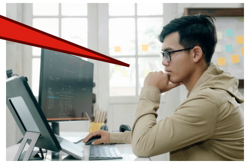
~~, integer~~

Act 3: Column Generation and the Cutting Stock Problem

Cut pattern	Quantity
17 17 17 17 17	0
31 31 31	0
38 38	0
45 45	43.75
38 38 17	201.5
31 31 38	197.5
17 17 17 45	0
45 38 17	9.5
	452.25

100

Solving the full master LP using column generation was easy to program using Gurobi's Python API



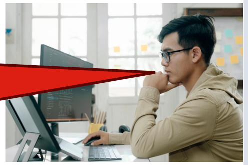
Yes, we use Gurobi for our routing problems. You could have programmed using C, C++, C#, Java and other languages as well.

In the last 30 years, LP/MIP solver APIs have made the implementation of decomposition algorithms like column generation much easier

Act 3: Column Generation and the Cutting Stock Problem

Cut pattern	Quantity	
17 17 17 17 17	0	
31 31 31	0	
38 38	0	
45 45	43.75	44
38 38 17	201.5	202
31 31 38	197.5	198
17 17 17 45	0	
45 38 17	9.5	10
	452.25	454
100		

Ah, now I see. Since this fractional solution satisfies demand, so does the integer solution obtained by rounding the fractional values up.

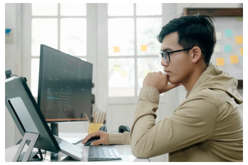


And the optimal LP objective value rounded up gives you a strong bound on the best possible integer solution.

Act 3: Column Generation and the Cutting Stock Problem

Cut pattern	Quantity	
17 17 17 17 17	0	
31 31 31	0	
38 38	0	
45 45	43.75	44
38 38 17	201.5	202
31 31 38	197.5	198
17 17 17 45	0	
45 38 17	9.5	10
	452.25	454
100		

And as long as the number of product roll types is small relative to the number of stock rolls cut, the rounded solution will have a very good MIP gap. In this case, with 4 product roll types, the worst case integer objective is 450 + 4 = 454 and the best case is 453



Intermission

Questions?

GUROBI
OPTIMIZATION

(Longer) Intermission

Let p1,…,pj and r1,…,rk be all extreme points and extreme rays of Ax=b, x>=0.   Reformulate the LP as A(u1 p1 + … + uj pj + v1 r1 + … + vk rk) = b with ui >= 0 and sum uj = 1….

But first, what are the takeaways from Act 3?

- The Column Generation formulation was much larger, but it was stronger than the compact formulation.
- The iterative procedure involving a dialog between the Boss and the Software Engineer that emerged in Acts 1 and 2 now has a corresponding mathematical interpretation involving a ”dialog” between the restricted master problem and sub problem. In both cases, information was traded that helped refine the best solution.
- The number of nonzero variables in an optimal basic solution to an LP is bounded by the number of constraints, not the number of variables.

GUROBI OPTIMIZATION

Column Generation Basics

Consider the LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & Dx \geq b \\ & x \geq 0 \end{aligned}$$

Complicating constraints

Easy constraints

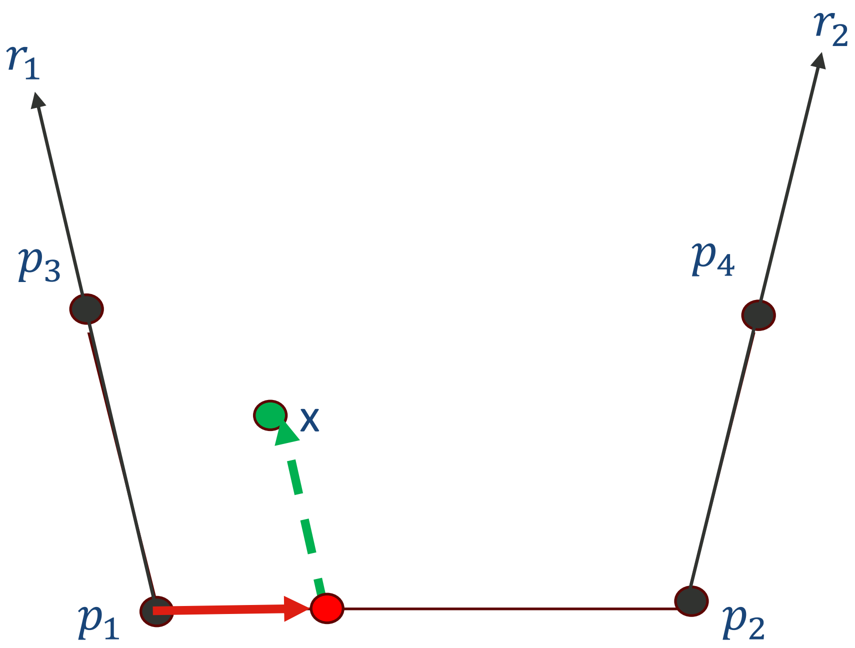
- To simplify the discussion, assume the feasible regions associated with the easy and complicating constraints are both bounded.
    - This just simplifies the math; it does not fundamentally alter the algorithmic computations.

Extreme points and extreme rays

- Any point in a polyhedron (feasible region of an LP) can be represented as a convex combination of extreme points and nonnegative linear combination of extreme rays.

Unbounded case

Bounded case



Let $p_1, p_2, \ldots p_k$ represent the extreme points (i.e., basic feasible solutions) of the easy constraints:

$\forall\, x \geq 0, Dx \geq b$:

$$x = \sum_{j=1}^{k} \lambda_j \, p_j + \sum_{i=1}^{l} \sigma_l \, r_l$$

$$\sum_{j=1}^{k} \lambda_j = 1$$

$$\lambda \geq 0, \sigma \geq 0$$

Column Generation Basics

Consider the LP

$$
\begin{aligned}
\min \quad & c^T x \\
s.t. \quad & Ax = b \\
& Dx \geq b \\
& x \geq 0
\end{aligned}
$$

Complicating constraints

Easy constraints

Let $p_1, p_2, \dots p_k$ represent the extreme points (i.e., basic feasible solutions) of the easy constraints:

$\forall\; x \geq 0, Dx \geq b$:

$$
\begin{aligned}
x = \; & \textstyle\sum_{j=1}^{k} \lambda_j \, p_j + \sum_{i=1}^{l} \sigma_l \, r_l \\
& \textstyle\sum_{j=1}^{k} \lambda_j = 1 \\
& \lambda \geq 0, \sigma \geq 0
\end{aligned}
$$

Extreme points

Extreme rays

Column Generation Basics

k can be huge, but let’s substitute the extreme point representation of the easy constraints into our original formulation.

$$\boxed{\begin{aligned} &\forall\, x \geq 0, Dx \geq b: \\ &x = \textstyle\sum_{j=1}^{k} \lambda_j\, p_j \\ &\qquad \textstyle\sum_{j=1}^{k} \lambda_j = 1 \\ &\qquad\qquad \lambda \geq 0 \end{aligned}}$$

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & Dx \geq b \\ & x \geq 0 \end{aligned}$$

→

$$\begin{aligned} \min \quad & c^T \textstyle\sum_{j=1}^{k} \lambda_j\, p_j \\ s.t. \quad & A \textstyle\sum_{j=1}^{k} \lambda_j\, p_j = b \\ & \textstyle\sum_{j=1}^{k} \lambda_j = 1 \\ & \lambda \geq 0 \end{aligned}$$

Column Generation Basics

Call this reformulation the full master problem

(Implicit) data

Variables

$$
\begin{aligned}
\min \quad & c^T \textstyle\sum_{j=1}^{k} \lambda_j \, p_j \\
s.t. \quad & A \textstyle\sum_{j=1}^{k} \lambda_j \, p_j = b \\
& \textstyle\sum_{j=1}^{k} \lambda_j = 1 \\
& \lambda \geq 0
\end{aligned}
\quad \Longleftrightarrow \quad
\begin{aligned}
\min \quad & \textstyle\sum_{j=1}^{k} \boxed{(c^T p_j)} \, \lambda_j \\
s.t. \quad & \textstyle\sum_{j=1}^{k} \boxed{(A \, p_j)} \, \lambda_j = b \\
& \textstyle\sum_{j=1}^{k} \lambda_j = 1 \\
& \lambda \geq 0
\end{aligned}
$$

Column Generation Basics

Master problem

(Implicit) data

Variables

$$
\begin{array}{lll}
\min & \sum_{j=1}^{k} \boxed{(c^T p_j)}\, \lambda_j & \\
s.t. & \sum_{j=1}^{k} \boxed{(A\, p_j)}\, \lambda_j = & b \\
 & \sum_{j=1}^{k} \lambda_j = 1 & \\
 & \lambda \geq 0 &
\end{array}
$$

Suppose A_2 has 10000 constraints and 10^{30} variables. At any simplex method iteration, how many variables are basic?

A) < 10000

B) 10000

C) More than 10000 but less than 10^{30}

D) 10^{30}

The master problem has a potentially huge number of variables. But as long as the number of constraints is modest, the number of variables in an optimal basic solution is modest. Can we avoid explicitly representing all the variables in the master problem?

Column Generation Basics

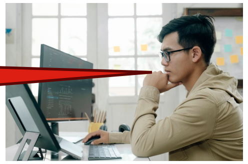
Can we avoid explicitly representing all the variables in the master problem?

$$Let\ J\ \subset K = \{1, \ldots, k\}\ be\ a\ small\ subset\ of\ the\ extreme\ points\ of\ the\ easy\ constraints$$

Restricted Master Problem:

$$
\begin{array}{ll}
\min & \sum_{j \in J} (c^T p_j)\, \lambda_j \\
s.t. & \sum_{j \in J} (A\, p_j)\, \lambda_j = b \\
 & \sum_{j \in J} \lambda_j = 1 \\
 & \lambda \geq 0
\end{array}
$$

Wait a minute. How do I know this problem isn't so restricted that it's infeasible?

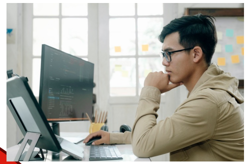


You don't. But in many cases a small collection of extreme points that is feasible can be easily constructed.

Column Generation Basics

Can we avoid explicitly representing all the variables in the master problem?

Let $J \subset \{1, \ldots, k\}$*be a small subset of the extreme points of the easy constraints*



Restricted Master Problem:

$$
\begin{aligned}
\min \quad & \sum_{j\in J} (c^T p_j)\,\lambda_j \\
s.t. \quad & \sum_{j\in J} (A\, p_j)\,\lambda_j = b \\
& \sum_{j\in J} \lambda_j = 1 \\
& \lambda \geq 0
\end{aligned}
$$

Remember how easily you constructed your first solution to the cutting stock problem? It wasn't that cost efficient, but it was feasible

Cut pattern	Quantity	Demand
17 17 17 17 17	43	211
31 31 31	132	395
38 38	305	610
45 45	49	97
100	529	

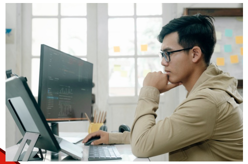
Column Generation Basics

Can we avoid explicitly representing all the variables in the master problem?

$Let\ J\ \subset \{1, \dots, k\} be\ a\ small\ subset\ of\ the\ extreme\ points\ of\ the\ easy\ constraints$

Restricted Master Problem:

$$\begin{aligned} \min \quad & \sum_{\boxed{j \in J}} (c^T p_j)\, \lambda_j \\ s.t. \quad & \sum_{\boxed{j \in J}} (A\, p_j)\, \lambda_j = b \\ & \sum_{\boxed{j \in J}} \lambda_j = 1 \\ & \lambda \geq 0 \end{aligned}$$



But even if you can’t do that, you can add a single auxiliary column to address the shortfalls or excesses associated with the variables in the restricted master problem

Column Generation Basics

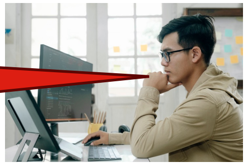
Can we avoid explicitly representing all the variables in the master problem?

Let $J \subset \{1, \ldots, k\}$ *be a small subset of the extreme points of the easy constraints*

Restricted Master Problem (RMP):

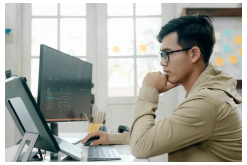
$$\begin{aligned} \min \quad & \sum_{\boxed{j \in J}} (c^T p_j)\, \lambda_j \\ s.t. \quad & \sum_{\boxed{j \in J}} (A\, p_j)\, \lambda_j = b \\ & \sum_{\boxed{j \in J}} \lambda_j = 1 \\ & \lambda \geq 0 \end{aligned}$$

OK, so after I have solved the RMP, what do I do next?



Think Column Generation and the first pass you made with the cutting stock problem. You looked at patterns that appeared to be wasteful and tried to replace them with less wasteful ones that would reduce the overall cost. An optimal basic solution provides reduced costs that you can use to automate what you did manually.

Column Generation Basics



Standard form LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

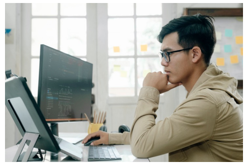
$$A = (A_B, A_N)$$

$$\bar{c}_j = c_j - c_B^T A_B^{-1} A_j = \boxed{c_j - (c_B^T A_B^{-1}) A_j} = \boxed{c_j - c_B^T (A_B^{-1} A_j)}$$

Column Generation Basics

Standard form LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & Ax = b \\ & x \geq 0 \end{aligned}$$



This is essentially what you did when you figured out how to reduce the cost from 529 to 486

Now you need to figure out how to automate that the process you did manually with the restricted master problem

Cost of increasing variable j

Adjustment to cost to reflect changes in basic variables

$$z_j = c_j - c_B^T \bar{A}_j$$

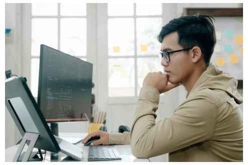
Cut pattern	Quantity	Change	Demand
17 17 17 17 17	$\boxed{0}$	-43	211
31 31 31	132		395
38 38	$\boxed{94}$	-211	610
45 45	49		97
38 38 17	211	211	
100	486	-43	

Column Generation Basics

$Let\ J \subset K = \{1, \dots, k\}\ be\ a\ small\ subset\ of\ the\ extreme\ points\ of\ the\ easy\ constraints$

Explicit columns of restricted master

Implicit columns of master



Master Problem:

$$
\begin{array}{ll}
\min & \sum_{j\in J} (c^T p_j)\,\lambda_j + \sum_{j\in K/J} (c^T p_j)\,\lambda_j \\
y: & \sum_{j\in J} (A\,p_j)\,\lambda_j + \sum_{j\in K/J} (A\,p_j)\,\lambda_j = b \\
\sigma: & \sum_{j\in J} \lambda_j + \sum_{j\in K/J} \lambda_j = 1 \\
 & \lambda \geq 0
\end{array}
$$

You have dual variables y from the RMP. But K/J has too many columns (extreme points) to compute all the reduced costs explicitly to find the one that reduces the overall cost the at the highest rate. But you can use the first reduced cost formula $z_j = c_j - y^T A_j$ to create a subproblem to efficiently find the most negative reduced cost.

Column Genera s

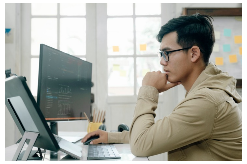
Explicit columns of restricted master

Implicit columns of master

Master Problem:

$$
\begin{array}{ll}
\min & \sum_{j\in J} (c^T p_j)\, \lambda_j + \sum_{j\in K/J} (c^T p_j)\, \lambda_j \\
y: & \sum_{j\in J} (A\, p_j)\, \lambda_j + \sum_{j\in K/J} (A\, p_j)\, \lambda_j = b \\
\sigma: & \sum_{j\in J} \lambda_j + \sum_{j\in K/J} \lambda_j = 1 \\
 & \lambda \geq 0
\end{array}
$$

You seek a column p that, when added to the RMP has a negative reduced cost, and is an extreme point of the easy constraints



Column

$$\begin{pmatrix} c^T p \\ A\, p \\ 1 \end{pmatrix}$$

Reduced cost

$$c^T p - y^T A\, p - \sigma = (c^T - y^T A)\, p - \sigma$$

c and A are data from the original LP
y and σ are optimal dual variables from the RMP
p is a vector of decision variables with dimension equal to the number of variables in the original LP

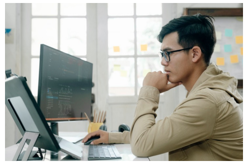
Column Generation Basics

Column

$$\begin{pmatrix} c^T p \\ A\, p \\ 1 \end{pmatrix}$$

Reduced cost

$$c^T p - y^T A\, p - \sigma = (c^T - y^T A)\, p - \sigma$$



Sub Problem:

$$\begin{aligned} &\min(c^T - y^T A)\, p \\ &s.t. \qquad Dp \geq b \\ &\qquad\qquad p \geq 0 \end{aligned}$$

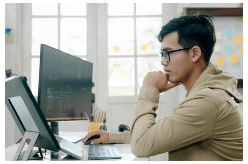
You figured out decision variables p yourself when you reduced the cost from 529 to 486. But you can automate this calculation by solving this subproblem instead.

c and A are data from the original LP
y and σ are optimal dual variables from the RMP
p is a vector of decision variables with dimension equal to the number of variables in the original LP

Column Generation Basics

Let p^* be an optimal solution to the Sub Problem:

$$\begin{aligned} &\min(c^T - y^T A)\,p \\ &s.t. \qquad Dp \geq b \\ &\qquad\qquad\ p \geq 0 \end{aligned}$$



If $c^T p^* - y^T A\,p^* - \sigma < 0$, then p^* is an extreme point of the easy constraints that can be added to the RMP.

Reoptimize the RMP with the added column and do another column generation iteration.

© 2024 Gurobi Optimization, LLC. Confidential, All Rights Reserved |

Column Generation Basics

Let p^* be an optimal solution to the Sub Problem:

$$\begin{array}{ll} \min(c^T \; - \; y^T A)\, p & \\ s.t. & Dp \geq b \\ & p \geq 0 \end{array}$$

Let $J \subset K = \{1, \dots, k\}$ be a small subset of the extreme points of the easy constraints

Explicit columns of restricted master

Implicit columns of master

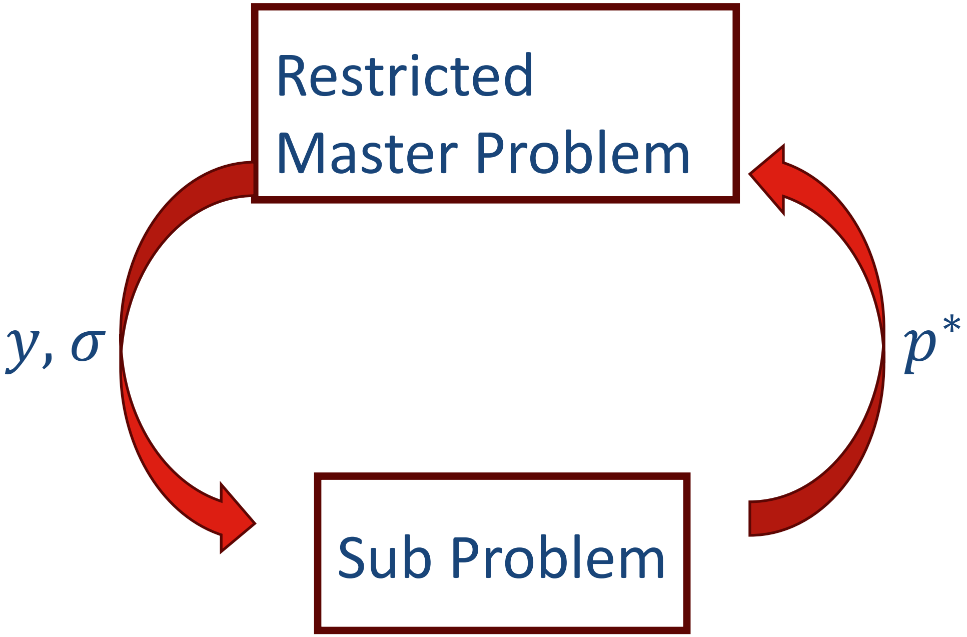
Master Problem:

$$\begin{array}{ll} \min & \sum_{j \in J} \; (c^T p_j)\, \lambda_j + \sum_{j \in K/J} \; (c^T p_j)\, \lambda_j \\ y: & \sum_{j \in J} \; (A\, p_j)\, \lambda_j + \sum_{j \in K/J} \; (A\, p_j)\, \lambda_j = \; b \\ \sigma: & \sum_{j \in J} \lambda_j + \sum_{j \in K/J} \lambda_j = 1 \\ & \lambda \geq 0 \end{array}$$

If $c^T p^* - \; y^T A\, p^* - \sigma \geq 0$, the Sub Problem solve proves that all implicit columns of the full master problem have even larger reduced costs, proving optimality of the full master problem in addition to the restricted master. The column generation algorithm terminates.

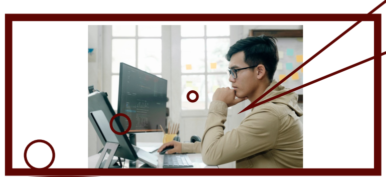
Column Generation Basics

Column generation is an iterative procedure involving a dialog between the restricted master and sub problems, similar to the dialog you had with your boss on the cutting stock problem.



529 sounds expensive. And look at all that wasted material.

Try this set of patterns that **costs** 486 instead of 529.



I can use this less wasteful pattern to ***reduce*** the usage of the most wasteful patterns

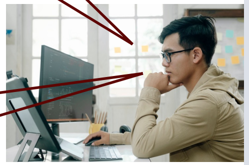
Intermission

Questions?

Act 5

That’s much better.   There’s much less waste.
This will make the operation solidly profitable.
How did you figure this out? Some new AI or ML technique?

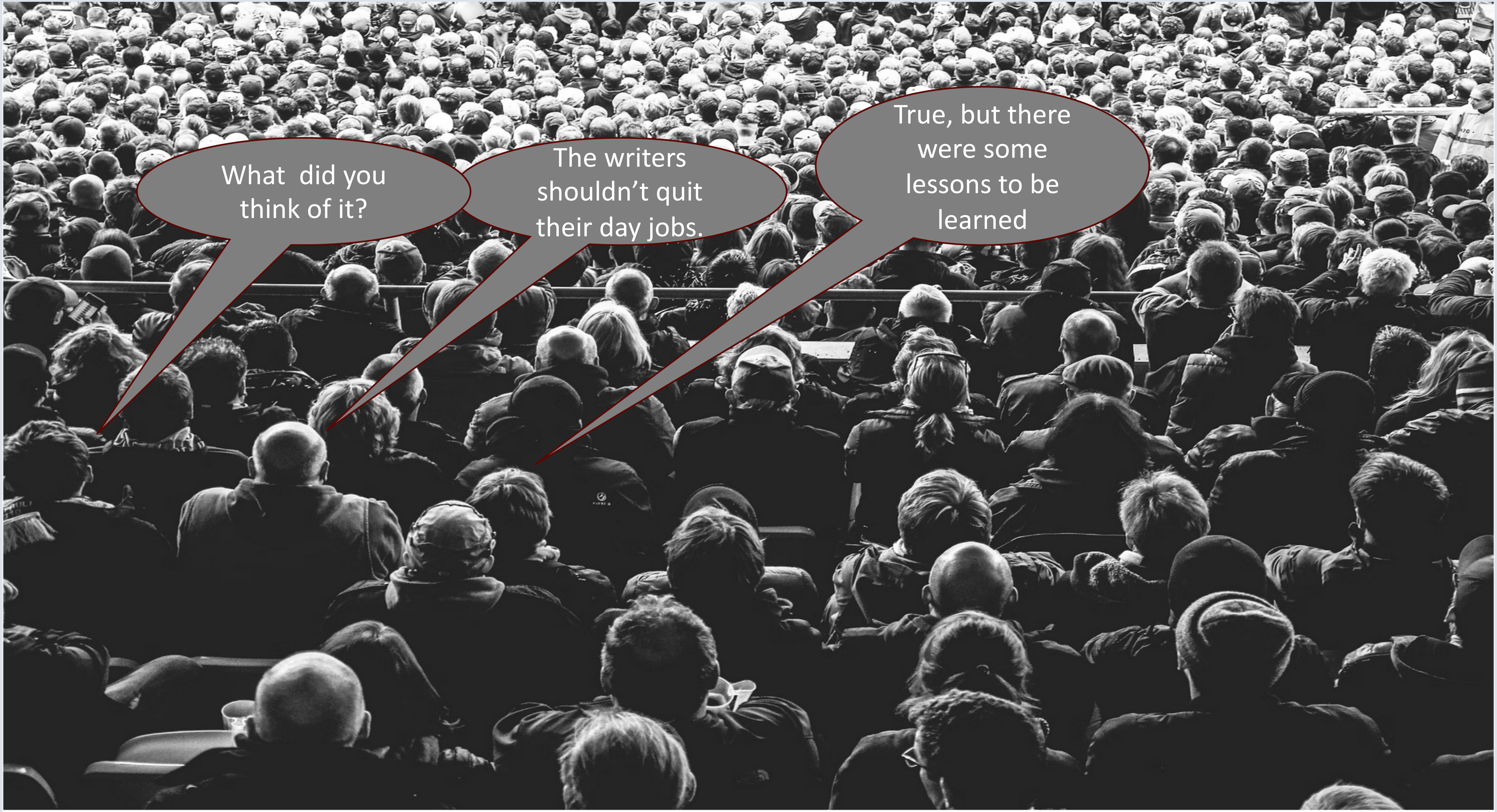
Here’s a solution that uses 454 stock rolls that is within 0.25% of optimal

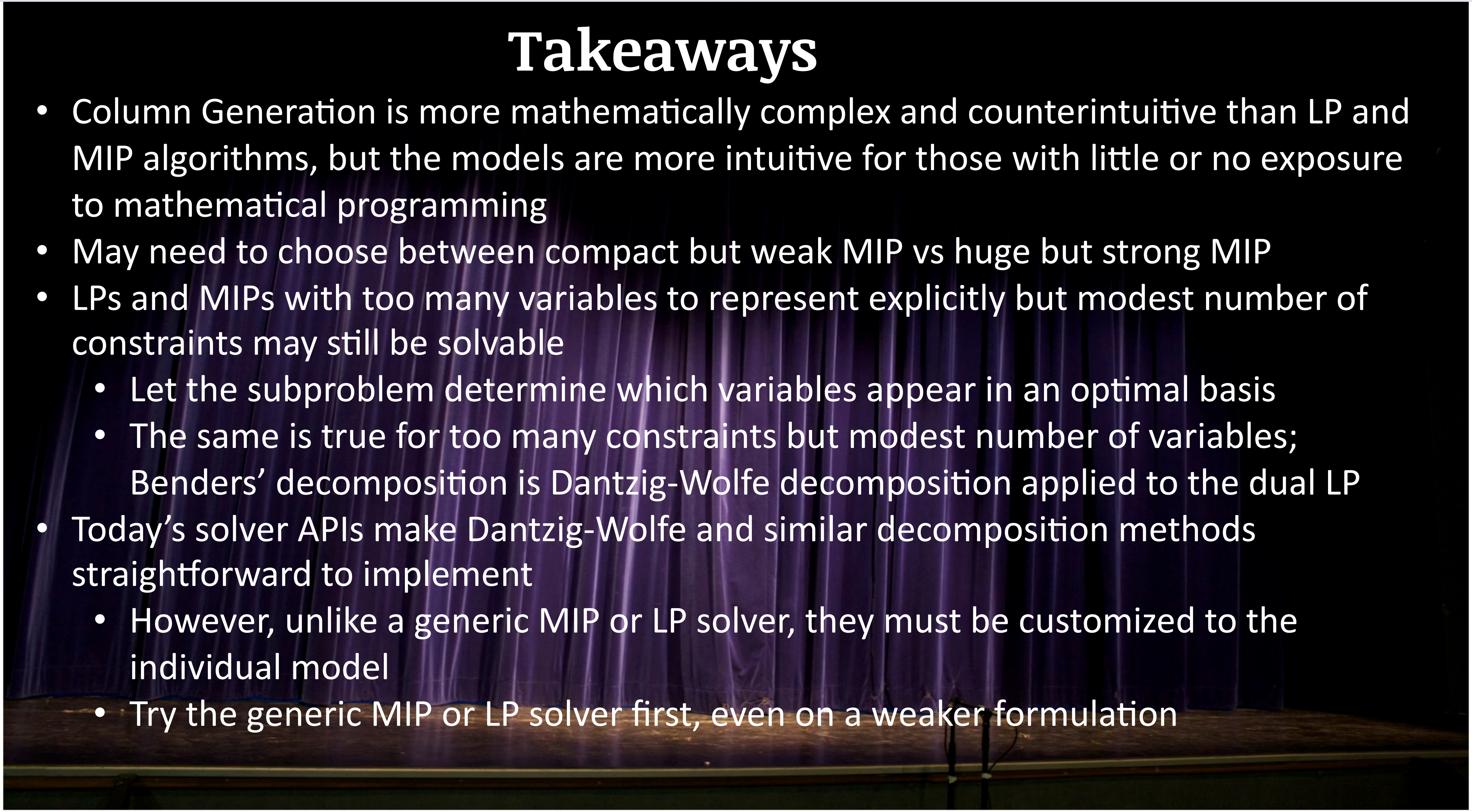


Using a technique that was invented in 1960 but was 30 years ahead of its time.

Cut pattern	Quantity
45 45	44
38 38 17	202
31 31 38	198
45 38 17	10
	454

The End


What did you think of it?
The writers shouldn’t quit their day jobs.
True, but there were some lessons to be learned



Takeaways

- Column Generation is more mathematically complex and counterintuitive than LP and MIP algorithms, but the models are more intuitive for those with little or no exposure to mathematical programming
- May need to choose between compact but weak MIP vs huge but strong MIP
- LPs and MIPs with too many variables to represent explicitly but modest number of constraints may still be solvable
  - Let the subproblem determine which variables appear in an optimal basis
  - The same is true for too many constraints but modest number of variables; Benders' decomposition is Dantzig-Wolfe decomposition applied to the dual LP
- Today's solver APIs make Dantzig-Wolfe and similar decomposition methods straightforward to implement
  - However, unlike a generic MIP or LP solver, they must be customized to the individual model
  - Try the generic MIP or LP solver first, even on a weaker formulation

GUROBI
OPTIMIZATION

Thank You

For more information: gurobi.com

Additional Resources

1. Marco E. Lübbecke, Jacques Desrosiers, (2005) Selected Topics in Column Generation. Operations Research 53(6):1007-1023. https://doi.org/10.1287/opre.1050.0234
2. Marco E. Lübbecke, Column Generation, Dantzig-Wolfe, Branch-Price-and-Cut. Video from CO@Work, 2020. https://www.youtube.com/watch?v=vx2LNKx48vY
3. Sergiy Butenko, Column Generation for the Cutting Stock Problem https://www.youtube.com/watch?v=O918V86Grhc
4. Sergiy Butenko, Dantzig-Wolfe Decomposition: Intro https://www.youtube.com/watch?v=IposxYVBUnY&t=891s
5. Video of an industrial strength paper cutting machine: https://www.youtube.com/watch?v=0zF6PWr7W8Y
6. Cutting stock (and other tech talk models) programs: https://github.com/Gurobi/techtalks/tree/main/mipformulations/programs/

Backup

Act 1

Hmm, not so fast. I can only use the pattern 97 times before generating surplus rolls of width 45

45	38	17

This looks cut pattern looks promising.   It has no wasted material at all.

Stock rolls

100

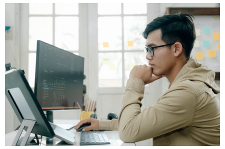
100

100

.
.
.

100

Product rolls	Demand
17	211 – 97 = 114
31	395
38	610 – 97 = 513
45	97



Act 1

Now I’m left with a new cutting stock problem with 3 product rolls and updated demands. Reducing the problem size is good, but I may be painting myself into a corner. And if I use the same greedy solution I used previously on this smaller problem, I only reduce the cost from 529 to 509. That won’t be enough to satisfy the boss. This is getting complicated. Let’s try another approach.

Stock rolls

100

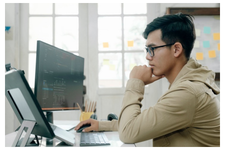
100

100

.
.
.

100

Product rolls	Demand
17	114
31	395
38	513



Question 4

Consider the LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & A_3 x = b \\ & x \geq 0 \end{aligned}$$

Suppose A_3 has 10000 constraints and 10^{30} variables. How many different bases are there?

A) 10000

B) $10^{30}!/10000!(10^{30} - 10000)!$

C) 10^{30}

D) Too many

Question 2

Consider the LP

$$\begin{aligned} \min \quad & c^T x \\ s.t. \quad & A_2 x = b \\ & x \geq 0 \end{aligned}$$

Suppose A_2 has 10000 constraints and 1000000 variables.

At any simplex method iteration, how many variables are basic?

A) < 10000

B) 10000

C) More than 10000 but less than 1000000

D) 1000000