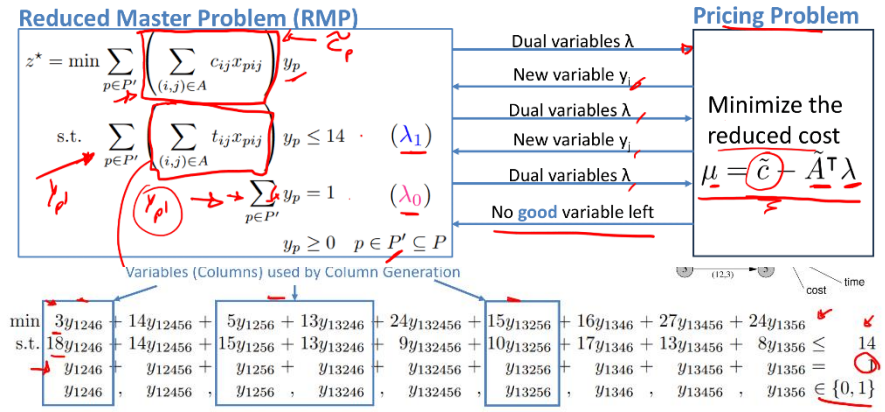
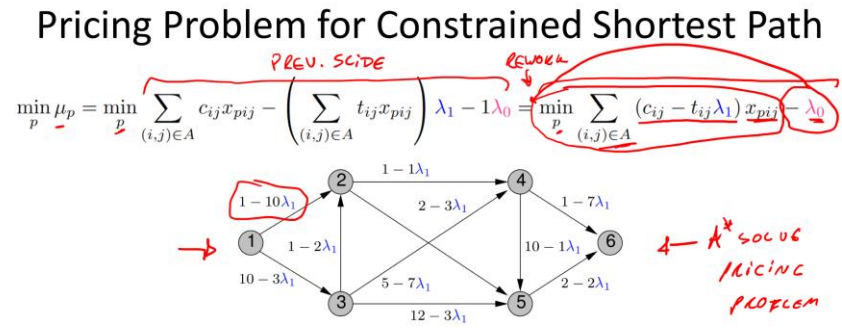
DECOMP

Column Generation: This technique addresses complicating constraints by reformulating the problem into a "simpler" problem with many variables, called the Master Problem -> iteratively builds a (RMP) by selecting which variables or columns to add. Applications: **Constrained Shortest Path Problem:** Find the minimum-cost path from a start node to a goal node in a network, with each arc representing a cost, solvable by Dijkstra's. Addition of complexity like a time limit, A* would require modifications to handle time constraints i.e. arrival times. **Flow Model**: Formulate the problem to push one unit of flow from the start node and extract it at the goal node, each arc (i,j) has a binary variable xij to denote if it is used in the path, converted into ILP and solvable by branch and bound. Complicating constraint that makes the problem challenging to solve directly, like the **time constraint** (not exceeding 14 units). **Reformulation Strategy**: Reformulate the problem by treating **paths** as individual variables. This approach shifts the focus to finding an optimal path without needing to consider all paths simultaneously. This separation enables us to introduce **paths dynamically as needed** and on demand generation. The constraint matrix is large, this approach leads to many variables (paths), especially in dense networks. However, we will use Column Generation to introduce only the paths that improve the solution, avoiding the need to consider all paths initially.



Reduced Master Problem (RMP): **current version of problem** with only a subset(P') of paths (columns). We repeatedly solve the RMP as more paths are added. In Column Generation, the **Pricing Problem** identifies paths that can enhance the current RMP solution by calculating their reduced cost. **Reduced Cost**: represents the **marginal increase in the objective** when each non-basic variable is marginally increased**.** Here roof(A) is the Basic part of A in roof(A)roof(x) = b for basic variables, and A~ is the non-basic part of A. c~ is the obj coeff of the NB vars, and lambda is the dual variables for the basic variables. **Lambda 0** won't affect our reduced cost, because we already have chosen one path, no need to incorporate, already been satisfied, and it's 1 if no path is chosen. Since the pricing problem does not have a comp cstr, it's easy to solve, and only paths with **negative reduced cost** are added to the RMP, as these are paths that could improve the objective function. **Tldr:** Start with a set of initial columns and solve the RMP using these columns. Then, use the dual variables from this solution to define a new subproblem known as the Pricing Problem. Solve the Pricing Problem to find the column with the minimum reduced cost. If this new column has a negative reduced cost, add it to the RMP and repeat the process. If not negative, then the optimal solution to the linear relaxation of the problem has been found. **Branch & Price:** If a path variable xij has a fractional value in the RMP solution x12=0.6), we create branches to enforce integer solutions. **Branch 1**: x12=0, **Branch 2**: x12=1, for 0, remove all paths that use 1->2, inverse for 1**. Very similar** process for **cutting stock**, that identifies new cutting patterns that minimize waste while



- Recall the reduced cost is: $\mu = \tilde{c} - \tilde{A}^T\lambda$
- For cutting stock problem, the reduced cost for a pattern j is: $\mu_j = 1 - \sum_i a_{ij}\lambda_i$

$$\min \sum_j x_j$$
$$\text{s.t.} \sum_j a_{ij} x_j \geq d_i \quad \forall i \quad (\lambda_i)$$
$$x_j \in \mathbb{N}_0$$

- We want to find a **feasible** pattern that minimises it:

$$\min_a 1 - \sum_i a_i\lambda_i \quad \text{s.t.} \sum_i w_i a_i \leq W, \quad a_i \in \mathbb{Z}_{\geq 0}$$

$$= 1 - \max_a \sum_i a_i\lambda_i \quad \text{s.t.} \sum_i w_i a_i \leq W, \quad a_i \in \mathbb{Z}_{\geq 0}$$

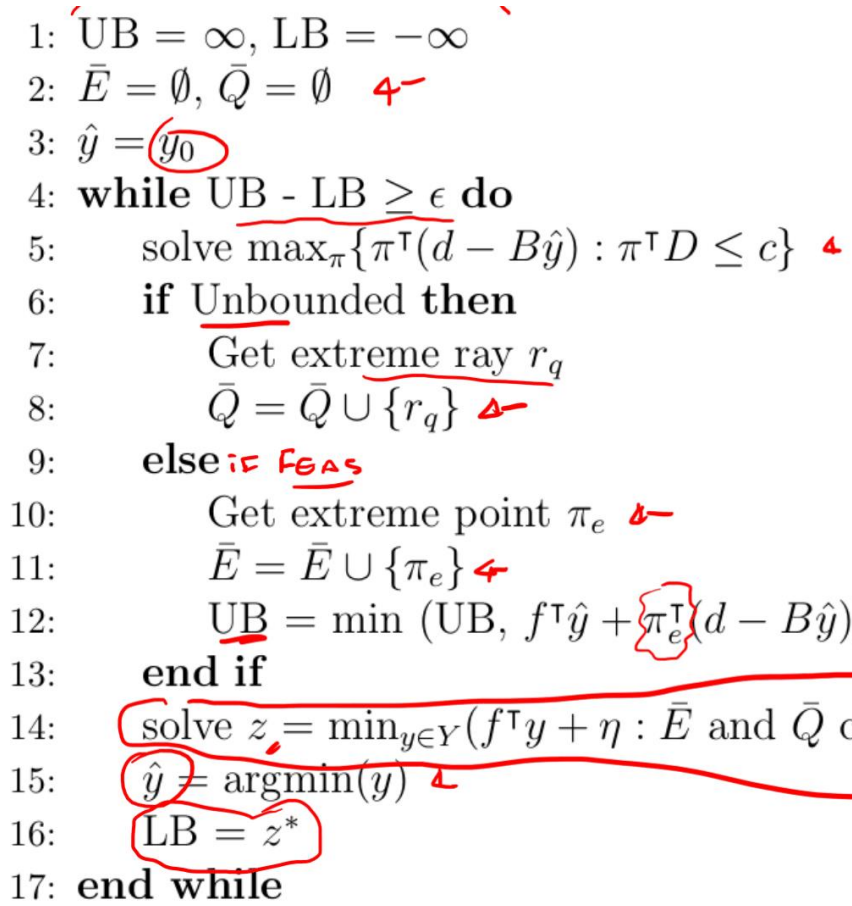
satisfying demand. Here a is the no of items of length i for pattern j. The max gives us a knapsack problem which is np easy. **Dantzig-Wolfe Decomposition** is a technique used to solve large-scale optimization problems with a **block structure** by breaking them into smaller, more manageable subproblems. Each subproblem, or "block," is optimized independently through **Column Generation**, where only the most promising solutions (columns) are generated. These subproblem solutions are then coordinated in a **Master Problem** that enforces global constraints, ensuring all blocks align to meet the overall objective. The process iterates, with new constraints added to the subproblems based on the Master Problem's solution, making it an efficient method for complex problems with strong interdependencies. **Complicating Constraints:** These constraints link variables in ways that make the feasible region difficult to define or compute efficiently, often adding complexity to the problem. Examples include the maximum time constraint in the constrained shortest path problem and the decision on cutting lengths in the cutting stock problem. **Complicating Variables**: These variables make solving the problem challenging because they are not continuous and/or link multiple constraints. Fixing their values simplifies the optimization problem. For instance, deciding whether to build facility k (yk∈{0,1}) in facility location problems.

nd**Benders Decomposition:** This technique handles complicating variables by decomposing the problem into an RMP and a subproblem. The RMP contains only the complicating variables and a new variable η. Its constraints involve only the complicating variables, optimality cuts, and feasibility cuts. Each iteration solves the subproblem by fixing the complicating variables to their RMP values, generating new cuts for the RMP. Bender's decomp, like colgen, works efficiently on block structures problems, where the subproblems (horizontal blocks) touch a little of the cvar. We don't know where the schools are, and the schools are the cvar i.e. y

X must be continuous; y can be continuous or integer. So, initialize y^, which is now a cons tant, so any constraint with y term is taken to the right side i.e. solve the prob in x for fixed y.

We like the DSP because y^ only appears in the obj, and cuts only apply on complicating constraints. Eita or η is the solution to the dual subproblem. If feasible, πe that achieves the maximum is a potential solution, and max of the dsp is atleast $\pi e\ (d - By)$. If DSP is unbounded, SP is infeasible.

η is going to be greater or equal than the objective fn of my dsp applied to this set of points -> this is going to be an optimality cut. If unbounded, rq(d-By)<=0 is the feasibility cut.



If the solution is unbounded, it is no longer a polytope but a polyhedron. In such cases, vertices alone are insufficient for representation, so we use vectors, known as extreme rays rq, to indicate directions in which the solution is unbounded. Dual variables and extreme rays belong to exponentially large sets but are generated only as needed. In **column generation**, we solve the linear relaxation by iteratively adding variables and typically use **branch and bound** to handle integer constraints. In contrast, **Benders' decomposition** operates independently of branch and bound, separating the problem into a master problem and subproblem and iterating to optimality through cuts, without requiring branching. **In LPs**, a complicating variable is the dual of a complicating constraint and vice versa. A complicating constraint in the primal LP can be tackled with Column Generation or addressed via Benders Decomposition in the dual LP. Busplus (application) - problem, where is the hub? Any bus stop potentially can be a hub, how do we connect these hubs, how do we minimize the no of hubs? Min of summation of xkbk and cik;xik, for cstr summation xik = 1, xik<=yk, yk is integer, xik is continuous.