Column Generation

- Number of variables (columns of A) huge
- Can *generate* column A_j by some "oracle" that can answer the question, "Does there exist a column with some property?" If so, the oracle returns one.

Example: Does there exist column with reduced cost < 0?

Modeling is like carpentry – we sometimes hit the nail right on the thumb.

Mark Twain (paraphrase)

A Column Generation Algorithm

1. Solve LP(J): min $\sum_{j\in J} c(j)x(j)$: $\sum_{j\in J} A(i, j)x(j) = b$, $x \geq 0$ for some $J \subseteq \{1, \ldots, n\}$
2. Using dual variables (π) that are optimal for LP(J), ask the oracle if there exists j ($\notin J$) such that $c(j) - \pi A_j < 0$
   If so, add it to J and perform pivot(s) to solve new LP(J).
   If not, we have optimal solution to LP over all columns.

This is pricing step of simplex method

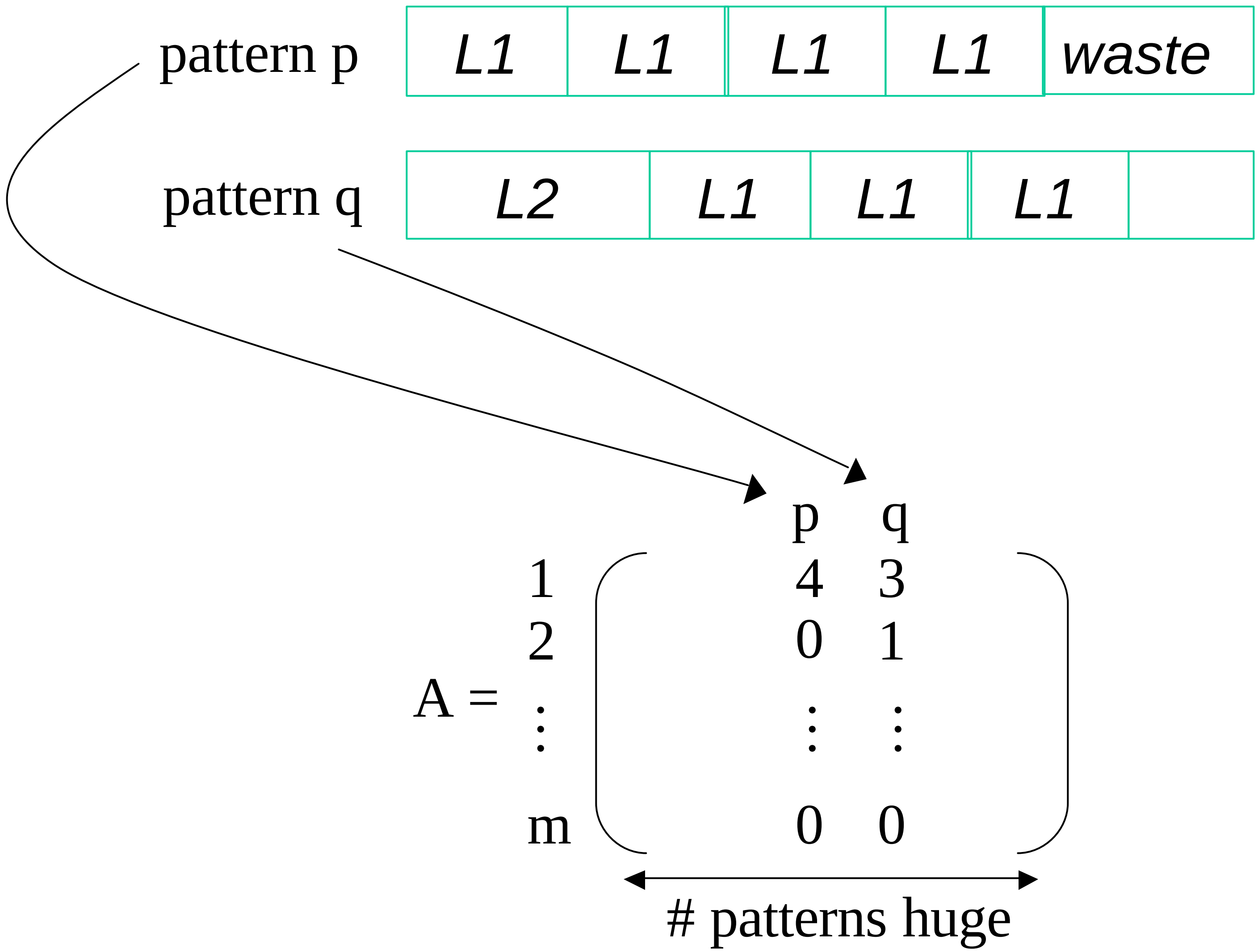
Cutting Stock Problem (the genesis)

Determine a way to cut standard sheets of material into various shapes (like clothes parts) to minimize waste.

Given: I = set of items
J = set of patterns
A(i, j) = number of times item i is in pattern j
b(i) = demand for item i
L = limit (length, area, volume, …)
Find: x(j) = number of times pattern j is used

$$\text{IP: } \min \Sigma_{j\in J}\, x(j): \quad \Sigma_{i\in I}\, A(i,j)x(j) \geq b(i),\ x \in Z^{+}$$

Column = Pattern



CG Subproblem = Knapsack

- We have (x, π) optimal over subset of patterns
- We want to price all patterns without explicitly generating them.

x optimal over all patterns $\leftrightarrow 1 - \pi A_j \geq 0$ for all $j \leftrightarrow \pi A_j \leq 1$ for all j

A_j represents a pattern $\leftrightarrow A(i, j)$ is in Z^+ and $\sum_{i \in I} L(i)A(i, j) \leq L$

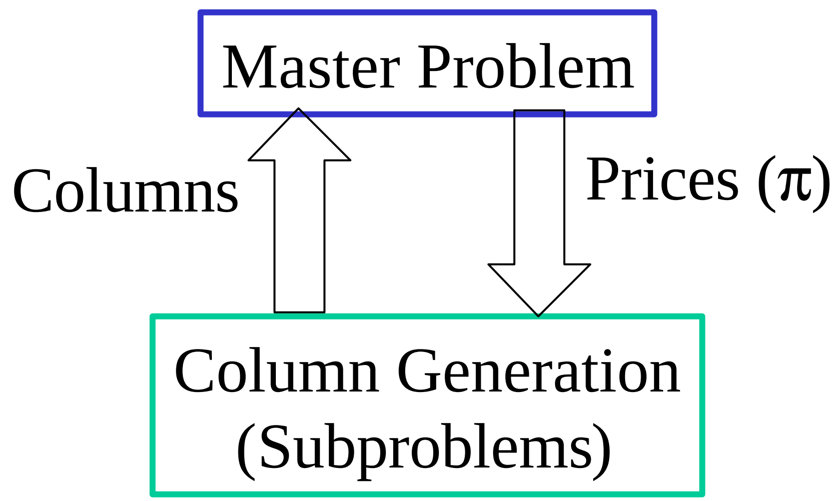
CGS: $\max \sum_{i \in I} \pi(i)N(i)$: $\sum_{i \in I} L(i)N(i) \leq L$, $N \in Z^+$

If $\pi N^* \leq 1$, (x, π) is optimal;
otherwise, N^* defines a new pattern that enters LP.

Column generation subproblem is a knapsack problem, which can be solved by dynamic programming.

CG Algorithm Structure

- Partition the MP into levels (Main/Subproblem; Master/Slave; Superior/Inferior)
- CG subproblem has structure that can be exploited

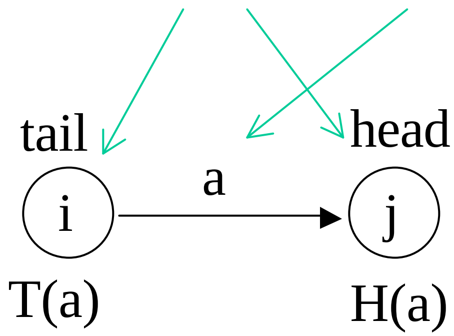


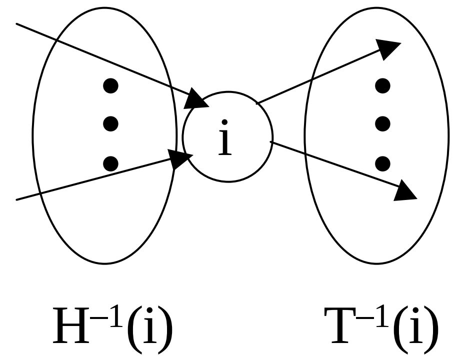
Cycle stops when primal-dual conditions are satisfied.

Multi-commodity Flows

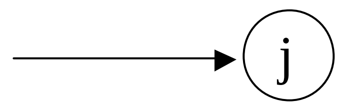
Network = nodes + arcs + data

Network notation:

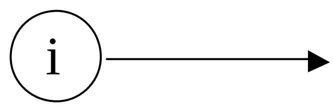




supply = tailess arc



demand = headless arc



data: costs, c(k, a), and capacities, u(k, a) & U(a)
b = [b(i)] supplies (sources) & demands (sinks)

size = n nodes, m arcs, K commodities

x(a, k) = flow of commodity k across arc a,
selected to optimize something (min cost, max flow)

Node-Arc Formulation

commodities share capacity

min $\Sigma_{k, a}$ c(k, a)x(k, a): x ≥ 0 (x integer),

arc capacity limits: x(a, k) ≤ u(a, k) and Σ_k x(a, k) ≤ U(a)

node flow balance: $\underbrace{\Sigma_{a\in T^{-1}(i)} x(a, k)}_{\text{flow out}} - \underbrace{\Sigma_{a\in H^{-1}(i)} x(a, k)}_{\text{flow in}} = b(i, k)$

LP size: mK variables, m + nK bundling constraints

other than simple bounds

Path-Cycle Formulation

$\mathcal{P}^k$ = set of paths from source node s(k) to sink node t(k) for commodity k

For P in $\mathcal{P}^k$:

f(P)	= flow on path P
I(a, P)	= indicator function for arc a in path P
C(k, P)	= unit cost of flow on path P in $\mathcal{P}^k$
	$= \Sigma_a I(a, P)c(k, a)$
d(k)	= demand for commodity k
$\mu(P)$	= upper bound on flow on P by commodity k
	= min{u(k, a): I(a, P)=1}

$$\min \Sigma_k \Sigma_{P \in \mathcal{P}^k} C(k, P)f(P): \;\; f(P) \geq 0$$

$$f(P) \leq \mu(P) \text{ and } \Sigma_k \Sigma_{P \in \mathcal{P}^k} I(a, P)f(P) \leq U(a)$$

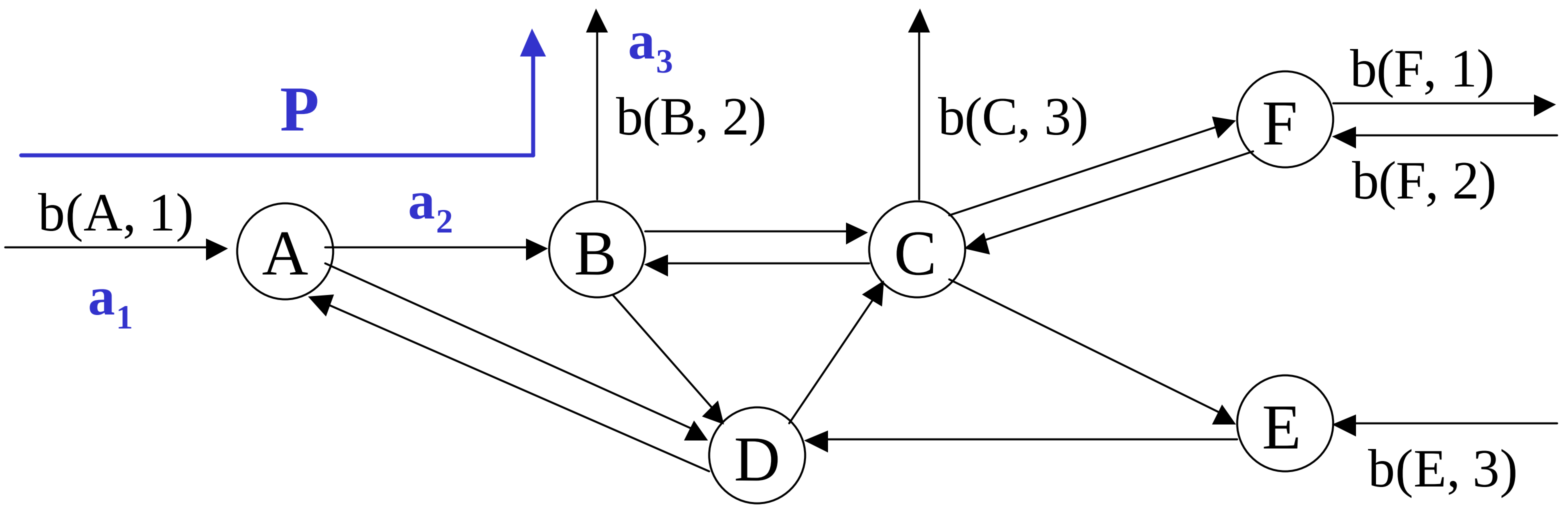
$$\Sigma_{P \in \mathcal{P}^k} f(P) = d(k)$$

Path-Cycle Formulation

P^k = set of paths from source node s(k) to sink node t(k) for commodity k

I(a, P) = indicator function for arc a in path P

$I(a_1, P) = I(a_2, P) = I(a_3, P) = 1; \; I(a_1, P) = 0$ for $i > 3$.



Comparisons

Node-Arc
- $m + nK$ bundling constraints
- mK variables

Path-Cycle
- $m + K$ bundling constraints
- $O(2^m)$ variables

Constraint structure enables CG:

$$x(a, k) = \sum_{P \in \mathcal{P}^k} I(a, P) f(P)$$

$$f(P) = \min_{a \in P} \Sigma_k \, x(a, k)$$

(apply recursively)

Column Generation Subproblem = Shortest Path

Using dual prices for current solution, reduced cost of path P

$= \Sigma_{a\in P}\big(c(k, a) + w(a)\big) - \pi(k)$

Dual variable of capacity constraint:

$\Sigma_k \Sigma_{P\in P^k} I(a, P) f(P) \le U(a)$

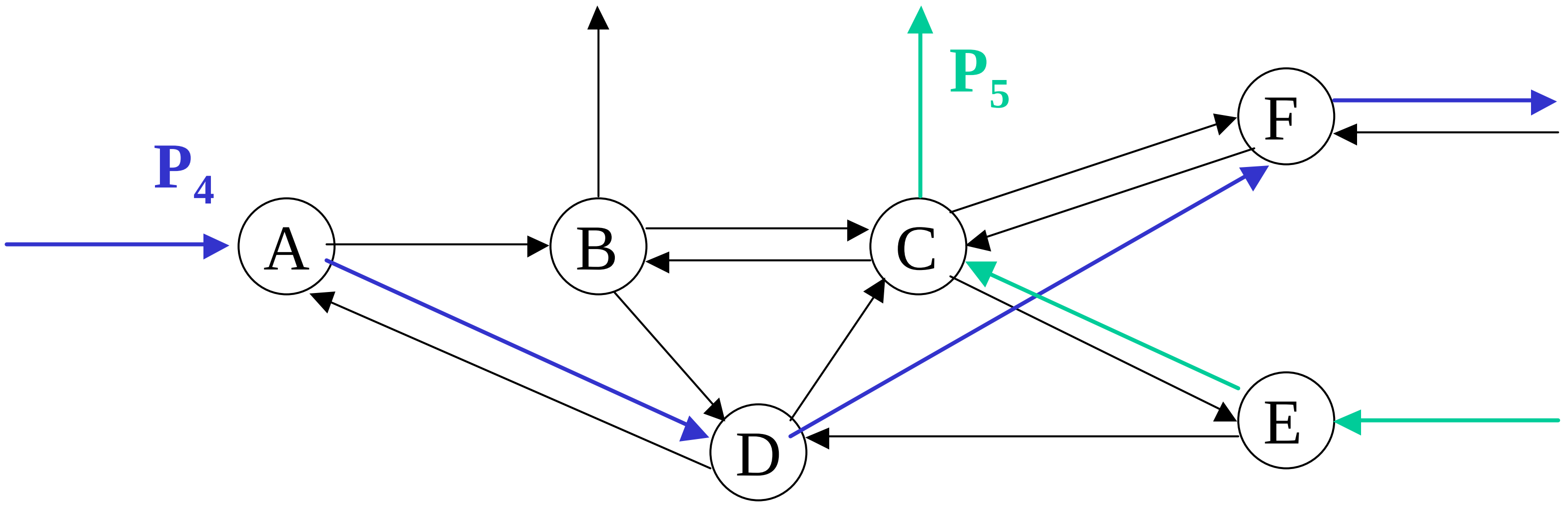
Dual variable of demand constraints:

$\Sigma_{P\in P^k} f(P) = d(k)$

Current solution optimal $\leftrightarrow \min\{\Sigma_{P\in P^k} \Sigma_{a\in P}\big(c(k, a) + w(a)\big)\} \ge \pi(k)$

column generation = seek path with lower cost to displace current flow

CGS: Shortest Path



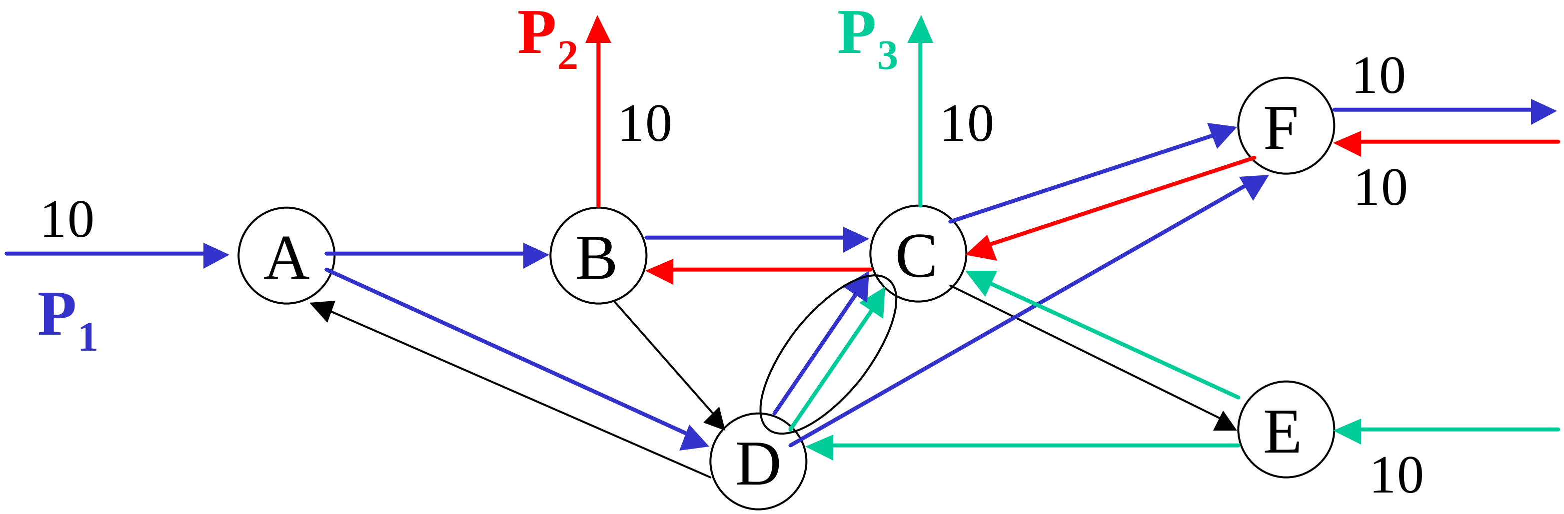
CGS for k=1 $\Rightarrow P_4$

CGS for k=2 $\Rightarrow$ no new path

CGS for k=3 $\Rightarrow P_5$

$c(a, k) = 1$ for all a, k

Expanded LP



$\mathcal{P}^1 = \{P_1, P_4\}$, $\mathcal{P}^2 = \{P_2\}$, $\mathcal{P}^3 = \{P_3, P_5\}$

U(a) = 10 for all a

$$\min 5f(P_1) + 4f(P_2) + 4f(P_3) + 4f(P_4) + 3f(P_5): \; f(P) \geq 0$$

$$f(P_t) \leq 10 \text{ for all } t$$

$$f(P_3) + f(P_4) \leq 10$$

for a = (D, C)

$$f(P_1) + f(P_4) = 10, \; f(P_2) = 10, \; f(P_3) + f(P_5) = 10$$

Flux Path Generation

- Single commodity
- Dynamic objective that favors paths with arcs that have not appeared many times in current paths

$$\min \Sigma_{P\in \mathsf{P}} C(P)f(P): \quad f(P) \geq 0$$

$$\Sigma_{P\in \mathsf{P}} I(a, P)f(P) \leq U(a)$$

$$\Sigma_{P\in \mathsf{P}} f(P) = d$$

$$C(P) = \Sigma_a I(a, P)c(a)$$

$$c(a) = \Sigma_t I(a, P_t)$$

CGS: Find shortest path using arc costs c(a)+w(a)

SVD Update

Given $\mathbf{P} = [P_1, \ldots, P_p] = U\Sigma V$, find SVD of $\mathbf{P}' = [\mathbf{P}\ \ P_{p+1}]$

☞ Several ways to do this, dating back to Bunch and Nielsen [1977]

Basic research is what I am doing when I don't know what I am doing.
Wernher von Braun

Column Generation for Extreme Pathway Analysis

- Start with some set of paths and find SVD of P
- Generate new path(s) with some property, ☛ **Research**
  or ascertain that no such path exists
- If new path is generated, update SVD and repeat

Everything should be made as simple as possible, but not simpler.
– Albert Einstein

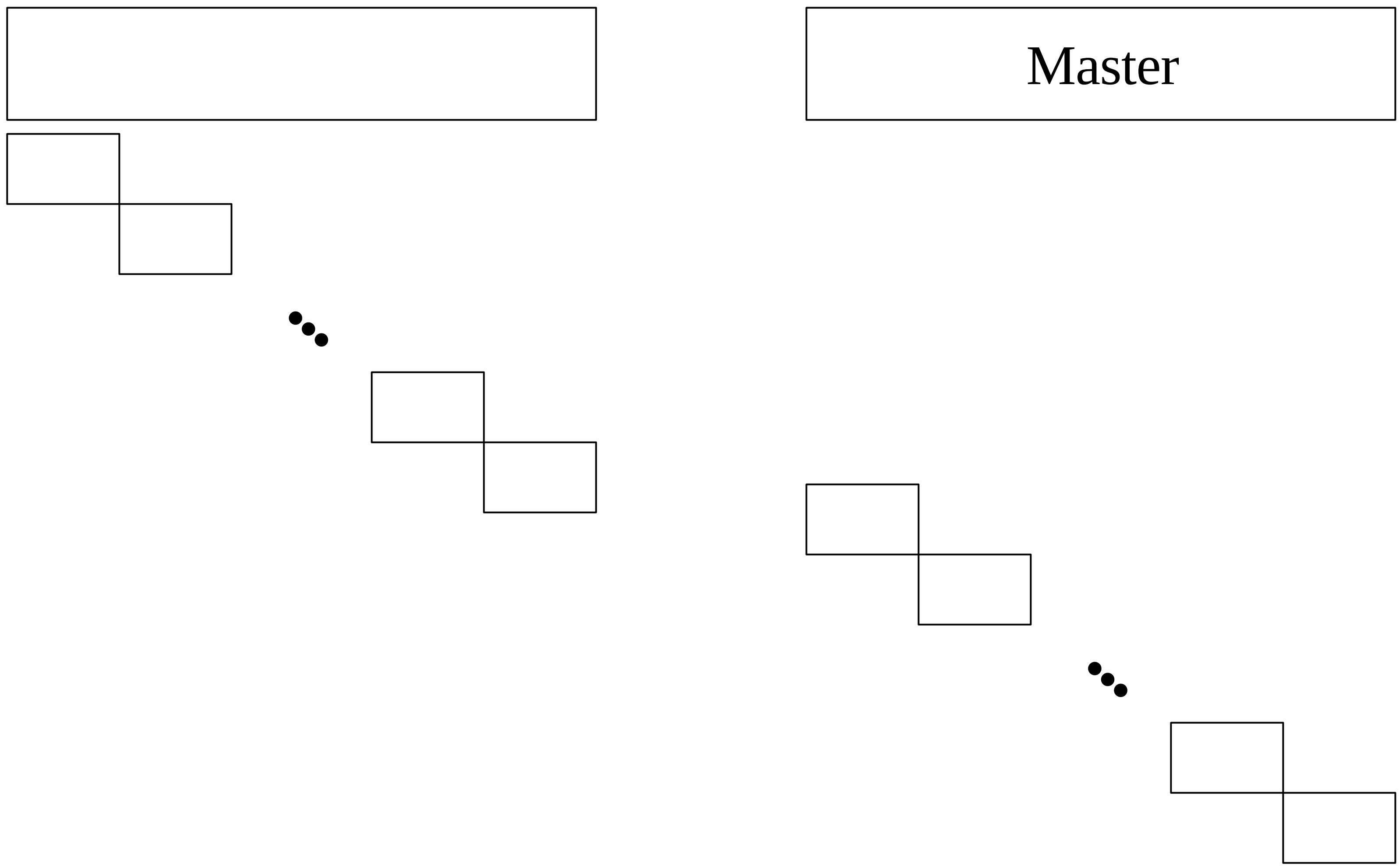
Dual = Separation Problem

- Solve LP with large number of rows (equations)
- First solve over restricted subset of rows (analogous to solving over subset of columns)
- Ask oracle if other rows are satisfied. If so, we are done; if not, ask the oracle to return a separating hyperplane that has current rows satisfied in one half space and a violation in the other.
- Rows in primal $\leftrightarrow$ columns in dual, so separation problem is dual of column generation problem.

A Separation Problem Algorithm

1. Solve LP(I): min cx: $A(i, .)\, x \geq b(i)$ for all $i \in I$ for some $I \subseteq \{1, \ldots, m\}$
2. Using the (primal) solution (x^*), find i for which $A(i, .)\, x^* < b(i)$. If none, we are done; if found, add to I and re-solve.

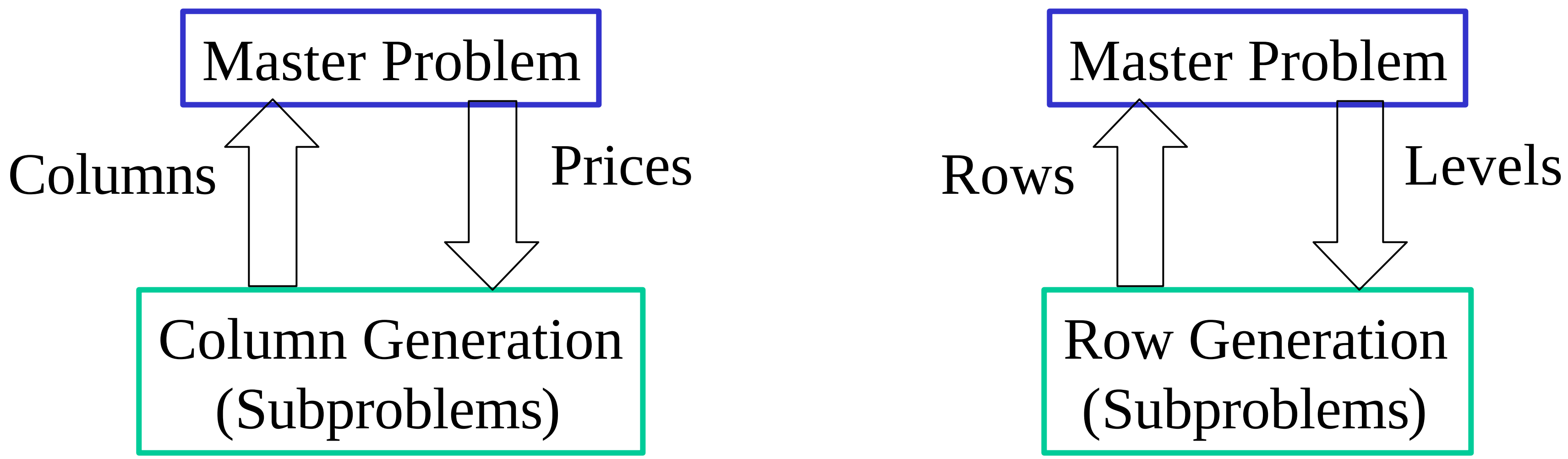
Block Diagonal Structure



Subproblems decompose

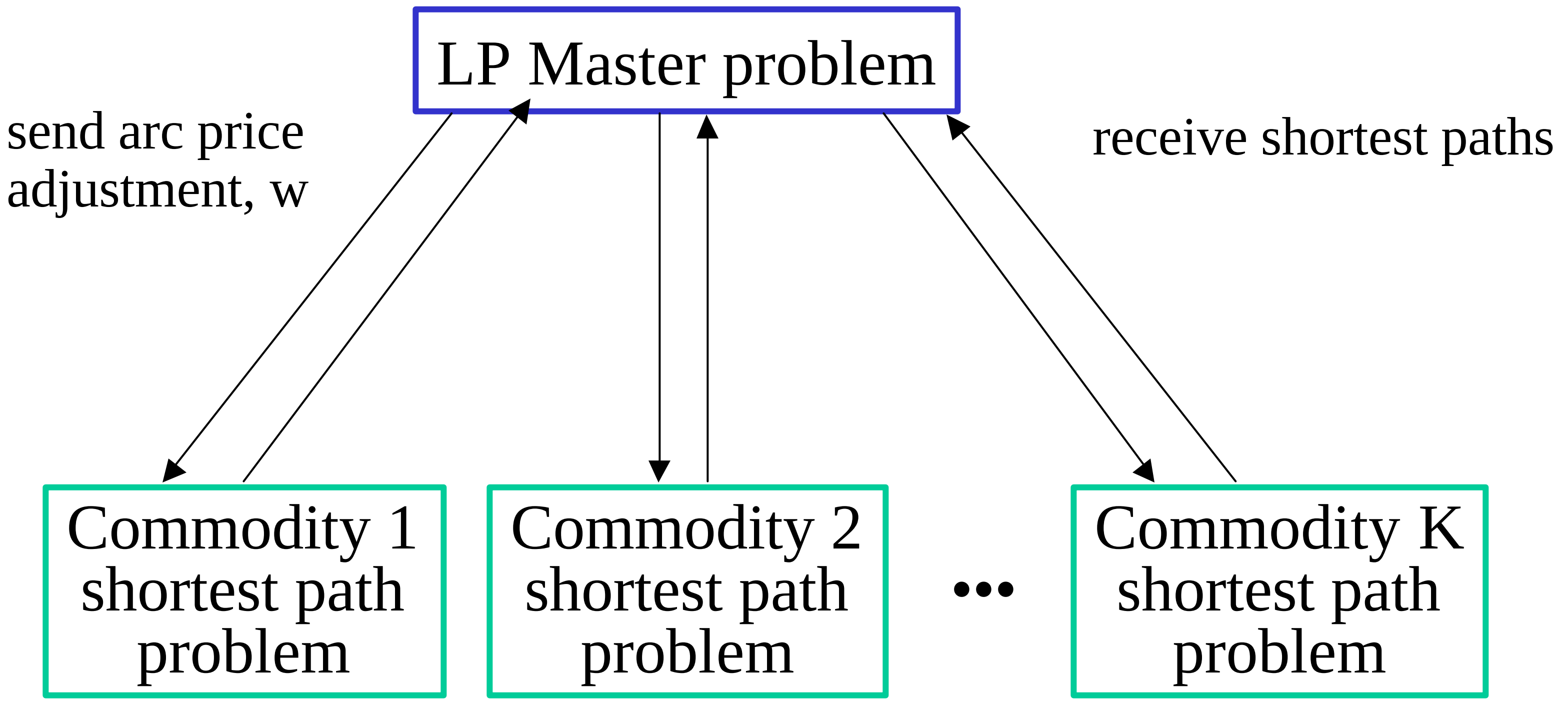
C/RG Algorithm Structure

- Partition the MP into levels (Main/Subproblem; Master/Slave; Superior/Inferior)
- Subproblem has structure that can be exploited (separable, network)



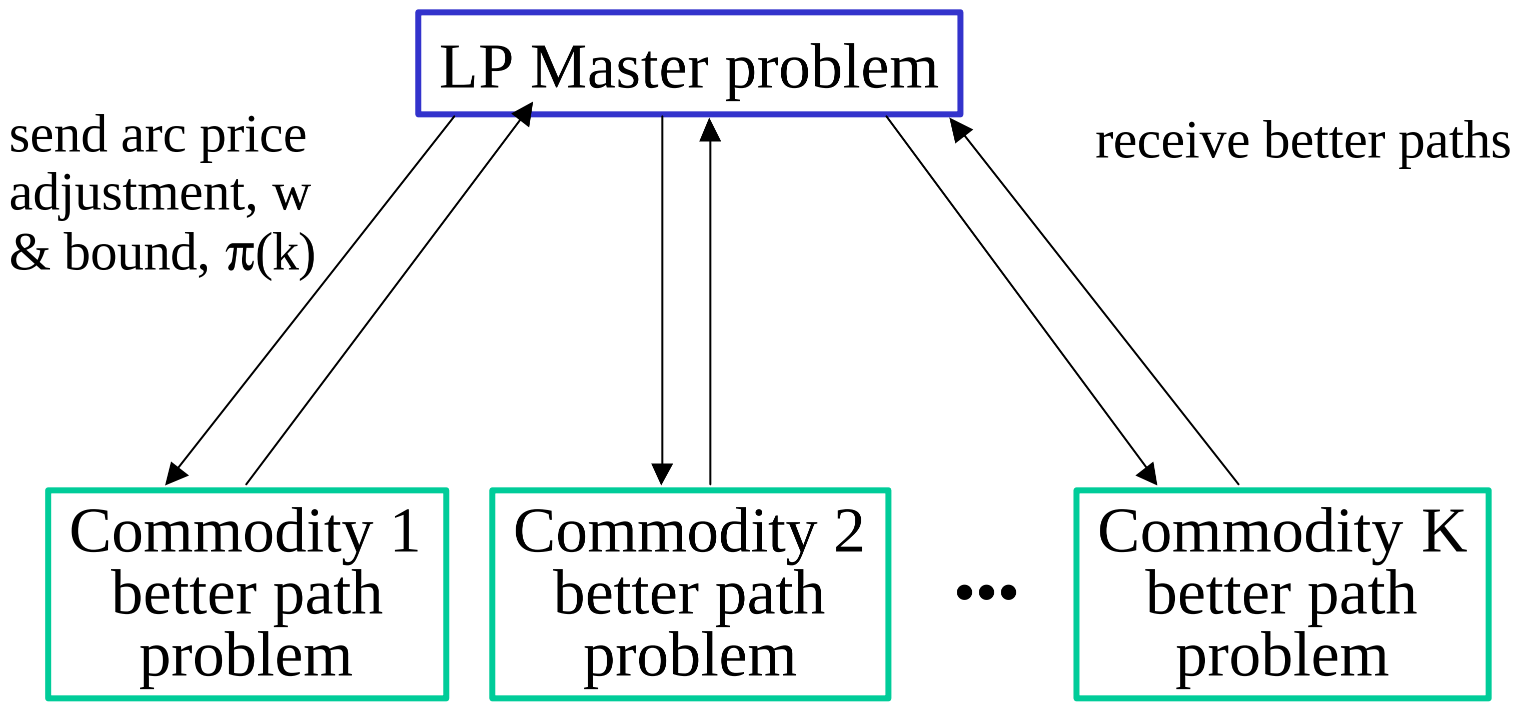
Cycle stops when primal-dual conditions are satisfied.

Dantzig-Wolfe Decomposition View of Multi-commodity Flow



Dantzig-Wolfe Decomposition View of Multi-commodity Flow

Modified to give bound info



Note potential for parallelization

Lagrangian Relaxation

$\max f(x): x \in X,\ g(x) \le 0$

$\max f(x) - \lambda g(x)$
$x \in X$

Two views:

1. Main problem is master & subproblem (Lagrangian) is column generator (Dantzig-Wolfe)
2. Lagrangian is master & subproblem is row generator (GLM)

This story can continue in NLP session.

References

R.K. Ahuja, T.L. Magnanti and J.B. Orlin, *Network Flows: Theory and Applications*, Prentice Hall, Englewood Cliffs, NJ, 1993.

G.B. Dantzig and P. Wolfe, Decomposition Principle for Linear Programs, *Operations Research* 8 (1960), 101-111.

L.R. Ford and D.R. Fulkerson, A Suggested Computation for Multi-commodity Flows, *Management Science* 5 (1958), 97-101.

P.C. Gilmore and R.E. Gomory, A Linear Programming Approach to the Cutting Stock Problem, *Operations Research* 9 (1960), 849-859.

M. Gu and S.C. Eisenstat, A Stable and Fast Algorithm for Updating the Singular Value Decomposition,

M.E. Luebbecke and J. Desrosiers, Selected Topics in Column Generation, Les Cahiers de GERAD G-2002-64, Group for Research in Decision Analysis, Montreal, Canada, 2002.