Branching for Branch and Price Algorithm

ISE Department
COR@L Lab
Lehigh University
zelihaakca@lehigh.edu

COR@L
COMPUTATIONAL OPTIMIZATION RESEARCH AT LEHIGH

01/26/07

Practical Issues: Branching Rules

- Branching must not destroy the structure of the subproblem.
- Branching should result child nodes that represent balanced sets of solutions which leads tighter bounds at each node.
- Original variable x is integer then, $\sum_{q \in Q} q\lambda_q$ be integer.
- Typically, branching on individual master variable λ_q results unbalanced tree and significant subproblem modifications.

1. Branching on and writing cuts in terms of master problem variables (λ).

.

2. Implicit branching through pricing problem.

.

3. Branching on and writing cuts in terms of original variables (x).

Master Problem

MP

$$\min \sum_{q \in Q} c_q \lambda_q$$

$$\text{s.t.}$$

$$\sum_{q \in Q} A_q \lambda_q \geq b \quad (\pi)$$

$$\sum_{q \in Q} \lambda_q \leq K \quad (v)$$

$$\lambda_q \geq 0 \quad \forall q \in Q$$

Pricing Problem

$$\min \; cy - \pi A y + v$$

$$Dy \geq d$$

$$y \in \mathbb{N}^{\ltimes}$$

Branching on λ

- Branching on fractional λ_q is not appropriate. Because:
  - significant changes for sub problem,
  - unbalanced branch and bound tree.
- If master problem solution $\lambda = (\lambda_1, .., \lambda_{|Q|})$ is fractional, then there exists $\hat{Q} \subseteq Q$ such that

$$\sum_{q \in \hat{Q}} \lambda_q = \alpha, \quad \alpha \text{ is fractional.}$$

- Then we can write the branching rule:

$$\sum_{q \in \hat{Q}} \lambda_q \leq \lfloor \alpha \rfloor \;\; \text{or} \;\; \sum_{q \in \hat{Q}} \lambda_q \geq \lceil \alpha \rceil$$

Branching on λ: Cont.

Generic master formulation with branching rules:

$$\min \sum_{q \in Q} c_q \lambda_q$$

s.t.

$$\sum_{q \in Q} A_q \lambda_q \geq b \quad (\pi)$$

$$\sum_{q \in Q} \lambda_q \leq K^j \text{ for } j \in G^u \quad (\mu_j)$$

$$\sum_{q \in Q} \lambda_q \geq L^j \text{ for } j \in H^u \quad (\nu_j)$$

$$\lambda_q \geq 0 \forall q \in Q$$

Reduced cost of the column:

$$\bar{c_q} = c_q - \sum_{i=1}^{m} \pi_i a_{iq} + \sum_{j \in G^u} \mu_j g_j(q) - \sum_{j \in H^u} \nu_j h_j(q)$$

- $g_j(q) = 1$ if column q has a nonzero coefficient in the row $j \in G^u$.
- $h_j(q) = 1$ if column q has a nonzero coefficient in the row $j \in H^u$.

Designing Branching Rules: Cont.

- Column Generation Subproblem:

$$\begin{aligned} \min \ \ cy - \pi Ay + \mu g - \nu h \\ Dy \geq d \\ g = g(y) \\ h = h(y) \\ y \in \mathbb{N}^{\ltimes} \\ g \in \{0,1\}^{|G^u|} \\ h \in \{0,1\}^{|H^u|} \end{aligned}$$

- $g = g(y)$, $h = h(y)$ are boolean functions: g=TRUE (=1) if generated column y will have a positive coefficient in the corresponding branching constraint.

Branching Based on Partitioning Q with a Hyperplane

Proposition

Given a feasible solution λ for master problem that is not integral, there exists a hyperplane $(\gamma, \gamma_0) \in \mathbb{Z}^{n+1}$ such that $\sum_{q \in Q : \gamma q \geq \gamma_0} \lambda_q$ is fractional.

- If master problem solution $\lambda = (\lambda_1, .., \lambda_{|Q|})$ is fractional, then there exists $(\gamma, \gamma_0) \in \mathbb{Z}^{n+1}$ such that

$$\sum_{q \in Q : \gamma q \geq \gamma_0} \lambda_q = \alpha, \quad \alpha \text{ is fractional.}$$

- The branching rule is

$$\sum_{q \in Q : \gamma q \geq \gamma_0} \lambda_q \leq \lfloor \alpha \rfloor \text{ or } \sum_{q \in Q : \gamma q \geq \gamma_0} \lambda_q \geq \lceil \alpha \rceil$$

Subproblem Modification

Let μ_j be the dual variable for $\sum_{q \in Q: \gamma q \geq \gamma_0} \lambda_q \leq \lfloor \alpha \rfloor$:

- Reduced cost of a column changed to:

$$\bar{c_q} = c_q - \sum_{i=1}^{m} \pi_i a_{iq} + \mu_j g_j(q)$$

  where $g_j = 1$ if column q satisfy $\gamma q \geq \gamma_0$.
- The objective function of subproblem is updated with $+\mu_j g_j$.
- Since it is unattractive for objective function, it is enough to put a constraint to force $g_j = 1$ when necessary.
- Constraint should be added to the subproblem to force $g_j = 1$ when the column, q satisfy $\gamma q \geq \gamma_0$.
-

$$(\gamma^j_{max} - \gamma^j_0 + 1) g_j \geq \gamma^j q - \gamma^j_0 + 1$$

  where $\gamma^j_{max} = max_{q \in Q} \gamma^j q$

Subproblem Modification: Cont.

Let v_j be the dual variable for $\sum_{q\in Q:\gamma q\geq\gamma_0}\lambda_q \geq \lceil\alpha\rceil$:

- Reduced cost of a column changed to:

$$\bar{c_q} = c_q - \sum_{i=1}^{m} \pi_i a_{iq} - v_j h_j(q)$$

  where $h_j = 1$ if column q satisfy $\gamma q \geq \gamma_0$.

- The objective function of subproblem is updated with $-v_j h_j$.
- Since it is attractive for objective function, it is enough to put a constraint to force $h_j = 0$ when necessary.
- Constraint should be added to the subproblem to force $h_j = 0$ when the column, q satisfy $\gamma q < \gamma_0$.
- $$(\gamma_0^j - \gamma_{min}^j)h_j \leq \gamma^j q - \gamma_{min}^j$$

  where $\gamma_{min}^j = min_{q\in Q}\gamma^j q$

Comments About the Rule

- Any fractional solution can be cut off.
- Number of possible sets is finite, the rule is finite.
- Complete branching rule.
- Not easy to find a hyperplane. Theoretical.
- In practice, consider hyperplanes with $\gamma = e^i$, but it does not guarantee the existence of hyperplanes.

Implicit Branching Through Pricing Problem

- Subproblem finds shortest path from an origin (s) to a destination (t) with minimum cost.
- Columns in master problem represent paths (arc incidence vectors).
- Solution of master problem is a combination of these paths satisfying the constraints.
- Let $q \in \{0,1\}^n$ be the column vector where n is the number of arcs in the network. If the solution to the master problem is not integral, then there exists an arc k such that the flow along the arc

$$\sum_{q \in Q : q^k = 1} = \lambda_q$$

is fractional.

Branching on the Pricing Problem

Branch 1

Flow on arc $k = (a \rightarrow b)$ is equal to 1.

- Master Problem: set $\lambda_q = 0$ for all $\{q \in Q\}$ if λ_q should be zero if arc k is in the solution.
- Pricing Problem: delete all arcs into b and from a except arc $a \rightarrow b$.

Branch 2

Flow on arc $k = (a \rightarrow b)$ is equal to 0.

- Master Problem: set $\lambda_q = 0$ for all $\{q \in Q : q^k = 1\}$.
- Pricing Problem: delete arc k.

Example from Cutting Strip Problem

- z_i^k= number of strips of width w_i cut from sheet k.
- $z_i^k = \sum_{q \in Q(k)} q_i^k \lambda_q$.
- Let z_i^k be fractional and $\lceil z_i^k \rceil = v$.
- Force

$$\sum_{q \in Q(k): q_i^k \geq v} \lambda_q \in \{0, 1\}$$

- In any cutting pattern for sheet k, there must be at least v strips of width w_i.
- In master problem, remove columns that do not satisfy the rule.
- In pricing problem, set a lower bound for q_i^k.
- Generic constraints are explained in Vanderbeck (2000).

Example from Cutting Strip Problem: Cont.

Symmetric Structure

- Forcing the rule for sheet k_1 $\rightarrow$ result columns for other sheets that do not satisfy the rule.
- Force:

$$\sum_{q \in Q : q_i \geq v} \lambda_q \text{ integer}$$

.

Choosing v

- Poorly chosen v results uneven partition of the solution space.
- Partition interval $[0, q_i^{max}]$ where q_i^{max} is the maximum value of q_i in any pattern.

Poggi and Uchoa (2003)[1] introduce explicit master:

Reformulation	Explicit Master	Pricing Problem
$\min\ cx$	$\min\ cx$	$\min -\pi x - v$
$x' - x = 0$	$Q\lambda - x = 0\ \ (\pi)$	$Dx \leq d$
$Ax = b$	$1\lambda = 1\ \ (v)$	$x \in \mathbb{Z}^n_+$
$Dx' \leq d$	$Ax = b\ \ (\mu)$	
$x', x \in \mathbb{Z}^n_+$	$\lambda, x > 0$	

.

- Size of explicit master is larger than DW master.
- LP relaxations are equal.
- Dual variable μ corresponds to $Ax = b$ is not used in pricing problem.
- Cuts in terms of x variables can easily be added to system $Ax = b$.

.

1.Poggi de Aragao, M., Uchoa, E.: Integer program reformulation for robust branch-and-cut-and-price. In: Annals of Mathematical Programming in Rio. Buzios, Brazil, 2003, pp.56.

Solving Subproblem

Assume we have N subproblems. Possible strategies:

- Solve N problems pick the best improving column to enter RMP.
- Add all columns with negative reduced cost to the RMP.
- Solve N problems sequentially, e.g. solve 1, then 2 , .., solve N.
- Solve the subproblems by selecting randomly.
- Solve subproblem heuristically to generate quick columns.
- Use column pool to keep generated columns.
- Delete columns with positive reduced cost from RMP.