Branch-and-price algorithms

J.M. Valério de Carvalho
vc@dps.uminho.pt

Departamento de Produção e Sistemas
Escola de Engenharia, Universidade do Minho
Portugal

Ciclo di Seminari 'Column Generation'
Metodi e Modelli per l'Ottimizzazione Combinatoria
Corso di Laurea Magistrale in Informatica
Dipartimento di Matematica Pura e Applicata
Università degli Studi di Padova
19^{th} - 28^{th} October 2011

Global outline

- Part I - Decomposition methods
- Part II - Applications
- Part III - Branch-and-price algorithms
- Part IV - Branch-and-price algorithms (cont.)
- Part V - Practical issues, stabilization, accelerating strategies and heuristics
- Bibliography

Part III

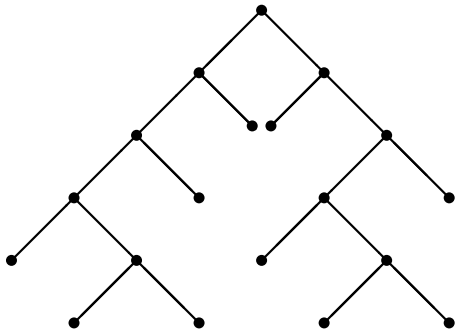
Branch-and-price algorithms

Outline

- Partition and branching
- Compatibility between master and sub-problem
- Coping with changes in sub-problem
- Application (binary variables): parallel machine scheduling

Getting integer solutions with branch-and-price

Branch-and-price = branch-and-bound + column generation



Methodology

- Branching constraints are introduced in the restricted master.
- After branching, deep in the tree, new columns may be needed.
- Column generation still has to work correctly.

Compatibility between Master Problem and Subproblem

Structure of the restricted master problem

- Branching constraints change the structure of the restricted master problem.
- Subproblem has to identify correctly the attractive and non-attractive columns with respect to the new structure.

Compatible (or *robust*) branching scheme

- Desirably, subproblem should be the same optimization problem both during the linear relaxation and branch-and-price.

Coping with changes

- Branching scheme should not induce intractable changes in the structure of the subproblem.

Branching schemes

Branching on variables of the reformulated model

- Regeneration of variables: a column set to zero by a branching constraint in the restricted master problem may turn out to be the most attractive column generated by the subproblem.

Branching on original variables

- Original variables: variables of model to which the Dantzig-Wolfe decomposition is applied.
- Successful in many applications.
- Often, original variables are related with flows in arcs.

A review of Partition and Branching

Each node of a branch-and-bound tree corresponds to a set of solutions obeying the constraints of the original problem and all branching constraints down to the node.

Partition:

- divides set of solutions into (desirably) mutually exclusive subsets,
- (should be a polynomial number of subsets),
- (desirably) corresponding to problems of the same type,
- eliminating the fractional solution of the node.

Basic and balanced branching schemes

- Basic branching rule: pick fractional x_{ij} and create 2 branches:

$$x_{ij} \leq \lfloor x_{ij} \rfloor$$
$$\text{and}$$
$$x_{ij} \geq \lceil x_{ij} \rceil$$

- Acting on a single variable may lead to a dive in the branching tree where no solutions will be found, and we still have to explore the other branch.
- Usually, better branching rules can be devised leading to more *balanced trees*, where we expect the subtrees to be of similar size.
- Branching schemes with unbalanced trees may perform very well on some instances, but very poorly on others.

Balanced branching schemes

Example 1: $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 1$

- Given the fractional solution $x_1 = 2/3$ and $x_6 = 1/3$,
- instead of using pair of branching constraints $x_1 \geq 1$ and $x_1 \leq 0$,
- use $x_1 + x_2 + x_3 \geq 1$ and $x_1 + x_2 + x_3 \leq 0$

Example 2: problems based on flows on arcs:

- For each node i, compute outflows $\sum_{j \in J} x_{ij}$
- Select set of successors $\overline{J}$: $\sum_{j \in \overline{J}} x_{ij}$ has a fractional value α,
- Use branching constraints:
  - $\sum_{j \in \overline{J}} x_{ij} \geq \lceil \alpha \rceil$
  - $\sum_{j \in \overline{J}} x_{ij} \leq \lfloor \alpha \rfloor$

Example

- left branch: $x_{12} + x_{13} \geq 2$
- right branch: $x_{12} + x_{13} \leq 1$

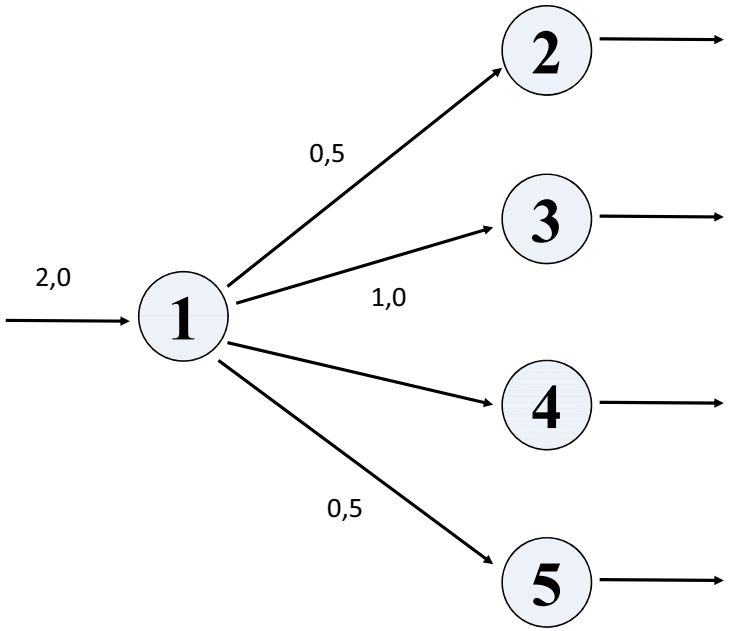


Figure shows part of a network

Selection of branching constraint

Branch first on decisions that have a larger impact on the solution.

Example 1: variable size bin-packing problem

- use a two-level branching scheme:
- branch first on bins until a bin integer solution is found, and then branch on items.
- in both levels, start with larger pieces (bins and items).

Branching in branch-and-price

Binary variables

- Ryan and Foster's rule for set partitioning problems
- Application: GAP
- Application: BCSP
- Application: multicommodity flow problems
- Application: vehicle routing

General integer variables

- general strategy
- adding branching constraints to the master problem explicitly
- Application: Cutting Stock Problems

Set partitioning problems

- S : finite set with m elements,
- $S_1, S_2, \ldots, S_n$, a collection of subsets of S.
- A partition of S is a collection of subsets, $S_{i_1}, \ldots, S_{i_j}, \ldots, S_{i_K}$, identified by $i_1, \ldots, i_j, \ldots, i_K$, such that:

$$\begin{aligned} \cup_{j=1}^{k} S_{i_j} &= S \\ S_{i_i} \cap S_{i_j} &= \emptyset, \ \forall i, j \end{aligned}$$

Ryan and Foster's rule (1981) for set partitioning problems

- Branches on variables of set partitioning problem:
- $\min\{cx : Ax = 1, x \in \{0,1\}^n\}$, and $A \in \{0,1\}^{m \times n}$.
- In the optimal integer solution of a set partitioning problem, each row is covered by ⋆exactly⋆ one column (variable).

Proposition

Given a fractional solution to $Ax = 1, x \geq 0$, *there are rows* r *and* s *such that* $0 < \sum_{j:a_{rj}=a_{sj}=1} x_j < 1$.

Many reformulated (column generation) models are set partitioning problems.

Branching rule

	x_{j_1}	x_{j_2}		
r	1	1	=	1
s	1		=	1

Branching rule: create two branches

- in left branch, rows r and s are covered by the same columns, *i.e.*, $\sum_{j:a_{rj}=a_{rs}=1} x_j = 1$.
- in right branch: rows r and s are covered by different columns, *i.e.*, $\sum_{j:a_{rj}=a_{rs}=1} x_j = 0$.

Example

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
1	1	1		1				$1 = 1$
2		1	1		1			$1 = 1$
3	1		1			1		$1 = 1$
	0.5	0.5	0.5					

For rows 1 and 2:

- left branch: $x_2 + x_7 = 1$
- right branch: $x_2 + x_7 = 0$

Dealing with branching constraints

	x_{j_1}	x_{j_2}		
r	1	1	=	1
s	1		=	1

A column j is feasible in the master problem,

- in left branch: if $(a_{rj} = a_{sj} = 1)$ or $(a_{rj} = a_{sj} = 0)$
- in right branch: if $(a_{rj} = a_{sj} = 0)$ or $(a_{rj} = 1, a_{sj} = 0)$ or $(a_{rj} = 0, a_{sj} = 1)$.

In the subproblem,

- in left branch: only accept solutions in which row 1 and 2 are both covered
- in right branch: if solution covers one row, the other must not be covered

Example: generalized assignment problem (GAP)

- maximize profit of assigning a set of jobs to agents with capacities $W_i, \forall i$.
- job j uses w_{ij} units of capacity of agent i.

$$\text{assignment variables } x_{ij} = \begin{cases} 1 & \text{, if job } j \text{ is assigned to agent } i \\ 0 & \text{, otherwise} \end{cases}$$

$$\begin{aligned} \max z &= \sum_{i=1}^{m} \sum_{j=1}^{n} p_{ij} x_{ij} \\ \text{subj. to} \quad & \sum_{j=1}^{n} w_{ij} x_{ij} \leq W_i, \ i = 1, \ldots, m \\ & \sum_{i=1}^{m} x_{ij} = 1, \ j = 1, \ldots, n \\ & x_{ij} \in \{0, 1\}, \ \forall i, j \end{aligned}$$

- Applications: in vehicle routing, resource scheduling, ...

Reformulated (set partitioning) model

- $K_i = \{x_1^i, x_2^i, \ldots, x_{k_i}^i\}$: set of all feasible assignments to agent i, $\forall i$.
- feasible assignment is a 0,1 vector $x_k^i = (x_{1k}^i, x_{2k}^i, \ldots, x_{nk}^i)$.

$$\text{ref. variables: } y_k^i = \begin{cases} 1, & \text{if feasible assignment } x_k^i \text{ is used for agent } i \\ 0, & \text{otherwise} \end{cases}$$

$$\forall i = 1, \ldots, m, \ k \in K_i.$$

$$\max z = \sum_{i=1,\ldots,m, \ k \in K_i} (\sum_{j=1}^{n} p_{ij} x_{jk}^i) y_k^i$$

$$\text{subj. to} \quad \sum_{i=1,\ldots,m, \ k \in K_i} x_{jk}^i y_k^i = 1, \ j = 1, \ldots, n$$

$$\sum_{k \in K_i} y_k^i \leq 1, \ i = 1, \ldots, m$$

$$y_k^i \in \{0,1\}, \ i = 1, \ldots, m, \ k \in K_i$$

- m knapsack subproblems, one for each agent.

Example with 5 jobs and 3 agents

	y_1^1	y_2^1	y_3^1	y_4^1	y_5^1	y_6^1	y_7^1	y_1^2	y_2^2	y_3^2	y_4^2	y_1^3	y_2^3	y_3^3	
job 1	1	1	1									1			$=1$
2	1			1				1	1			1			$=1$
3		1			1			1		1			1		$=1$
4			1		1	1			1	1	1		1	1	$=1$
5				1		1	1				1			1	$=1$
agent 1	1	1	1	1	1	1	1								≤ 1
2								1	1	1	1				≤ 1
3												1	1	1	≤ 1
max	7	8	6	5	4	9	5	6	4	6	4	5	3	5	

- set partitioning problem: instead of $\leq$ constraints, use "idle agent" columns (*i.e.*, slack variables for agent constraints).
- agent constraints are convexity constraints.

Branching rule [Savelsbergh'97]

Ryan and Foster's rule with one row belonging to a job and another row belonging to an agent $\Rightarrow$ branch on original variables x_{ij}.

Branching rule: create two branches

- in left branch $(x_{ij} = 1)$: agent i does job j
- in right branch $(x_{ij} = 0)$: assign job j to an agent i' other than i .

Branching constraints are not added explicitly to the Restricted Master Problem, but actions are taken to guarantee that the solution of a given node obeys the branching constraints imposed on the node.

Branching rule [Savelsbergh'97] (cont.)

left branch $(x_{ij} = 1)$:

- in master problem, set to 0:
  - all columns y_k^i of agent i that do not make job j (with $x_{jk}^i = 0$)
  - all columns $y_k^{i'}$ of agents i' other than i that make job j (with $x_{jk}^{i'} = 1$)
- in subproblem: always include job j in knapsack solution of agent i:

$$\begin{aligned} \max\ z = \quad & (\sum_{s \in S} \pi_s y_s) + \pi_j \\ subj.to \quad & \sum_{s \in S} w_s y_s \leq W_i - w_j \\ & y_s \in \{0,1\},\ \forall s \in S = \{1,\ldots,n\} \setminus \{j\} \end{aligned}$$

right branch $(x_{ij} = 0)$:

- in master problem: set to 0 all columns y_k^i of agent i that make job j
- in subproblem: exclude job j from knapsack subproblem of agent i.

Example: binary cutting stock problem (BCSP)

Binary cutting stock problem:

- demand constraints are disaggregated, and items are treated separately.
- only practical when quantities demanded by each client are very small, close to 1 unit.

$$
\begin{aligned}
min\ z_{IP} \quad &= \quad \sum_{j \in J} x_j \\
subj.\ to \quad & \quad \sum_{j \in J} a_{ij} x_j = 1,\ i = 1, 2, \ldots, m \\
& \quad x_j \in \{0, 1\},\ \forall j \in J
\end{aligned}
$$

- Special case of GAP when all agents are identical.

Example with 5 items

$W = 8$	cutting patterns						Demand b_i	
	x_1	x_2	x_3	x_4	x_5	x_6		
$w_i = 4$	1	1	1				$=$	1
4	1			1	1		$=$	1
3		1		1		1	$=$	1
2			1		1	1	$=$	1
2			1		1	1	$=$	1
min	1	1	1	1	1	1		

(some patterns may be missing...)

Set partitioning model:

- there are no convexity constraints (bins are equal).
- (convexity constraints would appear if bins were treated separately: same set of feasible solutions for every bin).
- no so structured as GAP.

Branching rule [Vance *et al.* '94]

Constraints are not added explicitly to the Master Problem.

in left branch: items r and s must belong to the same bin

- in Master Problem: combine rows r and s in a single row, and set to 0 all columns that are infeasible,
- in (knapsack) Subproblem: replace items r and s by a single item $m+1$ with $\pi_{m+1} = \pi_r + \pi_s$ and $w_{m+1} = w_r + w_s$.

$$\begin{aligned} \max\ z_s = \quad & \sum_{i \in S} \pi_i y_i \\ \textit{subj. to} \quad & \sum_{i \in S} w_i y_i \leq W \\ & y_i \in \{0,1\},\ \forall i \in S = \{1,\ldots,m\} \backslash \{r,s\} \cup \{m+1\} \end{aligned}$$

- subproblem generates columns with either both items r and s or none.

Branching rule [Vance *et al.* '94] (cont.)

in right branch: put items r and s in different bins

- in Master Problem: set to 0 all columns that are infeasible,
- in Subproblem: add extra constraint to avoid having both items r and s

$$\begin{aligned} \max\ z_s = \quad & \sum_{i \in S} \pi_i y_i \\ \textit{subj. to} \quad & \sum_{i \in S} w_i y_i \leq W \\ & y_r + y_s \leq 1 \\ & y_i \in \{0,1\},\ \forall i \in S \end{aligned}$$

- after b branchings, b pairs of constraints are added.

Structure of subproblem is preserved in GAP, but not in BCSP.

Binary multicommodity flow problem

Flow of K commodities, indexed by k, in a graph $G = (V, A)$
u_{ij} : capacity of arc (i, j)

Commodity k has a flow of q^k, from one unique supply node and to one unique demand node. Node i has:

a positive supply of b_i^k units of commodity k, if i is one of the origin nodes for k,
a positive demand of $-b_i^k$ units of commodity k, if i is one of the destination nodes for k,
a null value, otherwise.

The flow of each commodity must be routed in a single path.

Binary multicommodity flow problem: arc flow model

Decision variables $x_{ij}^k = \begin{cases} 1, & \text{if entire flow of commodity } k \text{ uses arc } (i,j) \\ 0, & \text{otherwise} \end{cases}$

c_{ij}^k : unit cost of arc (i,j)

$q^k c_{ij}^k$: cost of entire flow of commodity k in arc (i,j)

$$\begin{aligned} \min \quad & \sum_{k \in K} \sum_{(i,j) \in A} q^k c_{ij}^k x_{ij}^k \\ \text{subj. to} \quad & + \sum_{(i,j) \in A} x_{ij}^k - \sum_{(j,i) \in A} x_{ji}^k = b_i^k, \ , \ \forall i \in V, \forall k \in K \\ & \sum_{k \in K} q^k x_{ij}^k \leq u_{ij} \ , \ \forall (i,j) \in A \\ & x_{ij}^k \in \{0,1\}, \ \forall k, \forall (i,j) \in A \end{aligned}$$

Binary multicommodity flow problem: path model

P^k : set of paths between source node and destination node of commodity k.

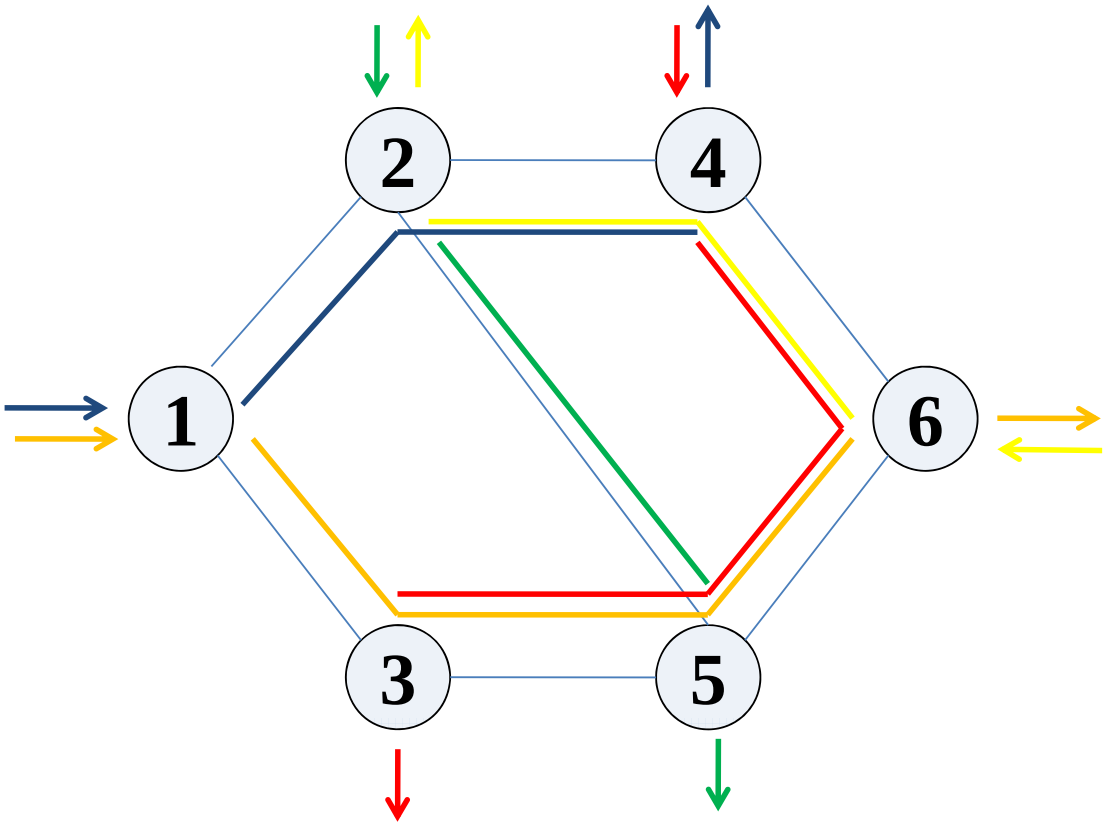
Reform. variables $y_p^k = \begin{cases} 1, & \text{if path } p \in P^k \text{ is used for commodity } k \\ 0, & \text{otherwise} \end{cases}$

c_p^k : corresponding unit cost for the path, *i.e.*, $c_p^k = \sum_{(i,j)\in A} \delta_{ij}^p c_{ij}^k$.

where $\delta_{ij}^p = \begin{cases} 1, & \text{if arc } (i,j) \text{ belongs to path } p \\ 0, & \text{otherwise} \end{cases}$

$$\begin{aligned} \min \quad & \sum_{k\in K} \sum_{p\in P^k} q^k c_p^k y_p^k \\ \text{subj. to} \quad & \sum_{k\in K} \sum_{p\in P^k} \delta_{ij}^p q^k y_p^k \le u_{ij} \ , \ \forall (i,j) \in A \\ & \sum_{p\in P^k} y_p^k = 1, \ \forall k \in K \\ & y_p^k \in \{0,1\}, \ \forall p \in P^k, \forall k \in K \end{aligned}$$

Example



(Starting) solution shown in Example

	y_1^{blue}	y_1^{orange}	y_1^{green}	y_1^{red}	y_1^{yellow}		
$(1,2)$	10					$\leq$	10
$(1,3)$		7				$\leq$	12
$(2,4)$	10				11	$\leq$	32
$(2,5)$			8			$\leq$	8
$(3,5)$		7		5		$\leq$	12
$(4,6)$				5	11	$\leq$	16
$(5,6)$		7		5	11	$\leq$	24
1	1					$=$	1
2		1				$=$	1
3			1			$=$	1
4				1		$=$	1
5					1	$=$	1
min	20	14	16	10	22		

Branching on variables of reformulated model

Binary multicommodity flow problem: branching on paths

- in left branch, $y_p^k = 1$ is easy to deal with:
  - commodity k is done,
  - just reduce the capacities of arcs in path p by q^k.
- in right branch, $y_p^k = 0$ forbids commodity k to use path p:
  - (shortest path) subproblem must know that it should not generate path p for commodity k,
  - which may be (and often is) the most attractive path to the subproblem after the branching constraint is added to the master problem ...

Avoiding the regeneration of variables ... to avoid a deadlock

Modify subproblem, so as to:

reject the most attractive column (if you do not want it in the master problem), the 2^{nd}, the 3^{rd}, ..., the k^{th} best columns, until an column acceptable is found.

Application: binary multicommodity flow problem:

use k^{th} best shortest path problem in subproblem.

Application: cutting stock problem:

Degraeve, Schrage' 99 : (modified) knapsack subproblem receives a list of forbidden solutions, and only generates acceptable solutions.

Binary multicommodity problem: branching on x_{ij}^k variables

x_{ij}^k are variables of the original model

- in right branch, $x_{ij}^k = 0$ forbids commodity k to use arc (i,j) : just remove arc (i,j) from (shortest path) subproblem graph.
- in left branch, $x_{ij}^k = 1$ forces commodity k to use arc (i,j) :
  - easy if there is a single constraint,
  - complicated if shortest path must go through a set of arcs, when several constraints are enforced in the node, deep in the tree.

Branching rule [Barnhart *et al.* '00]

Binary multicommodity flow problem: exclude set of arcs of commodity in one branch and the complementary set in the other branch

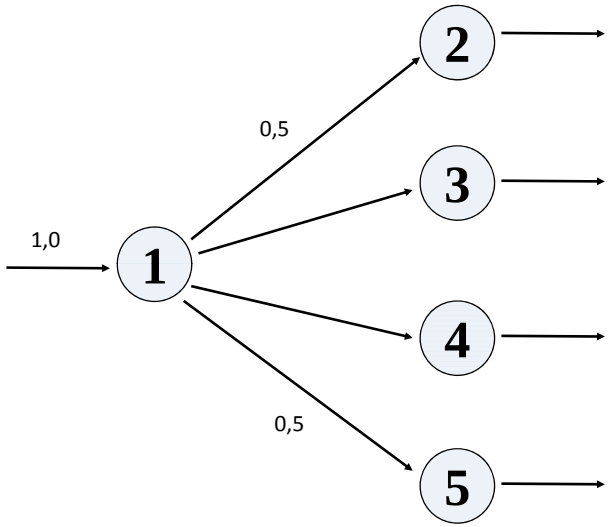
For a commodity k with fractional flows x_{ij}^k out of a node i:

- Choose a(n even) partition of the set J of successors of $i : \overline{J}$ and $J \backslash \overline{J}$,
- such that $\sum_{j \in \overline{J}} x_{ij}^k < 1$,
- and use branching constraints:
  - $\sum_{j \in \overline{J}} x_{ij}^k \leq 0$
  - $\sum_{j \in J \backslash \overline{J}} x_{ij}^k \leq 0$

Constraints are easy to enforce in the subproblem in both branches, because arcs are just removed from subproblem.

Example

- left branch: $x_{12}^k + x_{13}^k = 0$ – arcs $(1,2)$ and $(1,3)$ excluded
- right branch: $x_{14}^k + x_{15}^k = 0$ – arcs $(1,4)$ and $(1,5)$ excluded



left and right branches are not mutually disjoint:

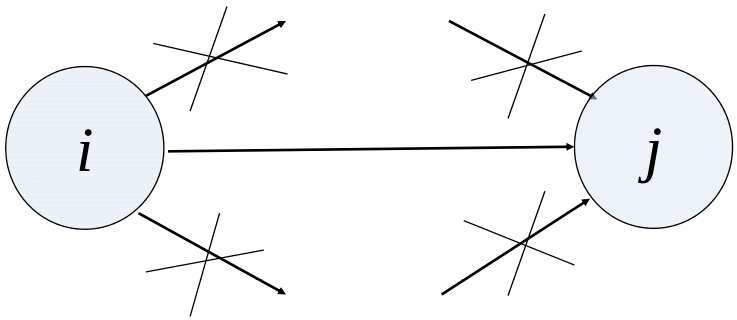
Solutions with null flow in all arcs $(1,2),(1,3),(1,4)$ and $(1,5)$ belong to both branches.

Vehicle routing with TW [Desrochers *et al.* '92]

All clients are visited once:

in left branch: cover clients i and j with the same route

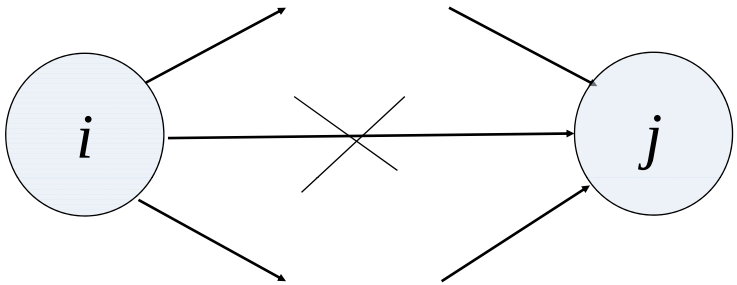
- in Subproblem network:
  - fix arc (i,j) at 1.
  - arcs $(i,k), k \neq j$ are removed
  - arcs $(l,j), l \neq i$ are removed
- in Master Problem: penalize all columns of master problem that use arcs removed in subproblem (penalty should be sufficient to drive then to 0)



Vehicle routing with TW [Desrochers *et al.* '92] (cont.)

in right branch: cover clients i and j with different routes

- in Subproblem network:
  - arc (i,j) is removed
- in Master Problem: again penalize all columns of master problem that use arcs removed in subproblem



Carpaneto and Toth's rule (1980)

- Pick a fractional variable y_r of the master problem corresponding to a route with s arcs: $v_1, v_2, \ldots, v_s, v_1$.
- Create $s+1$ branches on arc variables of the route:

$$
\begin{array}{lcl}
\text{branch } 1 & : & x_{v_1 v_2} = 0 \\
\text{branch } 2 & : & x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 0 \\
 & \ldots & \\
\text{branch } s & : & x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_{s-1} v_s} = 1,\ \ldots,\ x_{v_s v_1} = 0 \\
\text{branch } s+1 & : & x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_{s-1} v_s} = 1,\ \ldots,\ x_{v_s v_1} = 1
\end{array}
$$

- creates a polynomial number of branches
- used in J. Desrosiers, F. Soumis, M. Desrochers, Routing with Time Windows by Column Generation, Networks, 14, pp. 545–565, 1984.

Example: route with 3 arcs

Consider route: $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_1$.

$$
\begin{aligned}
&\text{branch } 1: && x_{v_1 v_2} = 0 \\
&\text{branch } 2: && x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 0 \\
&\text{branch } 3: && x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 0 \\
&\text{branch } 4: && x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 1
\end{aligned}
$$

	$x_{v_1 v_2}$	$x_{v_2 v_3}$	$x_{v_3 v_1}$
	0	0	0
branch 1	0	0	1
	0	1	0
	0	1	1
branch 2	1	0	0
	1	0	1
branch 3	1	1	0
branch 4	1	1	1

Dealing with branching constraints

branch 1 : $x_{v_1 v_2} = 0$:

- remove arc $x_{v_1 v_2}$ from subproblem

branch 2 : $x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 0$:

- group trips 1 and 2 both in the master problem and in subproblem
- remove arc $x_{v_2 v_3}$ from subproblem

branch 3 : $x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 0$:

- group trips 1, 2 and 3 both in the master problem and in subproblem
- remove arc $x_{v_3 v_1}$ from subproblem

branch 4 : $x_{v_1 v_2} = 1,\ x_{v_2 v_3} = 1,\ x_{v_3 v_1} = 1$:

- route is fixed: remove trips 1, 2 and 3

Considering branching constraints explicitly

In all the previous examples, the branching constraints were not added to the Restricted Master Problem.

General strategy [Vanderbeck, Wolsey'96]:

- Consider the structure of the RMP at a given node of the Branch-and-Price tree,
- Use dual information from the branching constraints in the subproblem.

Considering branching constraints explicitly (cont.)

Restricted master problem at a given node of the branch-and-price tree:

$$
\begin{aligned}
\min z \quad = \quad & \sum_{j \in J} c_j x_j \\
\text{s. to} \quad & \sum_{j \in J} a_j x_j = b \\
& \sum_{j \in J} x_j \leq U \\
& \sum_{j \in J} x_j \geq L \\
& x_j \geq 0, \text{ and integer, } \forall j \in J,
\end{aligned}
$$

$\pi \in \mathbb{R}^m, \mu \in \mathbb{R}_-, \nu \in \mathbb{R}_+$ are the dual variables corresponding to each set of constraints, respectively.

Considering branching constraints explicitly (cont.)

Subproblem:

- there are new constraints in the Restricted Master Problem,
- it may be necessary to consider additional binary variables in the subproblem to enforce in the subproblem the dual information of the Restricted Master Problem,

Compatibility:

- if that happens, subproblem loses its structure,
- it may become a general Integer Programming Problem.

Concluding remarks

- compatibility is a crucial issue in branch-and-price
- in models with binary variables, it is often possible to implement branch-and-price without adding explicitly the constraints.