Kernelization and Approximation for Finding a Perfect Phylogeny from Mixed Tumor Samples

Author: Wen-Horng, Sheu

Source: unpublished

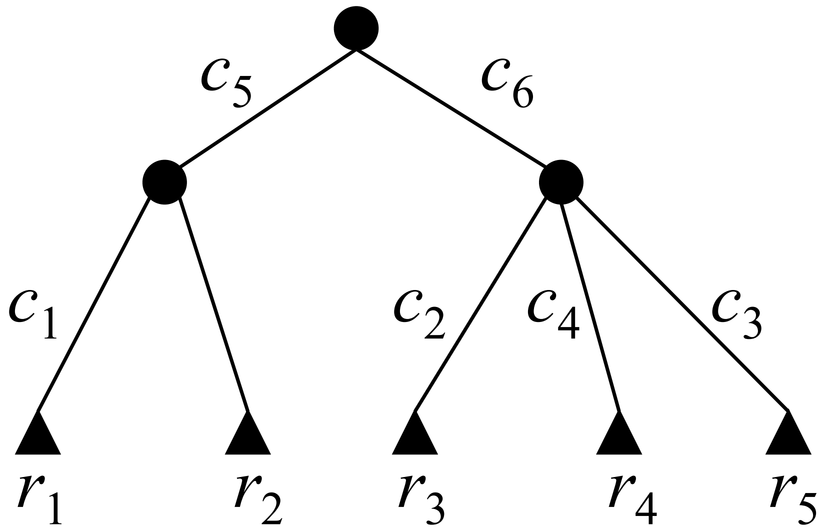
Speaker: Wen-Horng, Sheu

Outline

- **Introduction**
- Preliminaries
- A kernelization algorithm for MSRP
- An approximation algorithm for MSRP (**skipped**)
- Approximation algorithms for MDCRSP
- Conclusion and future work

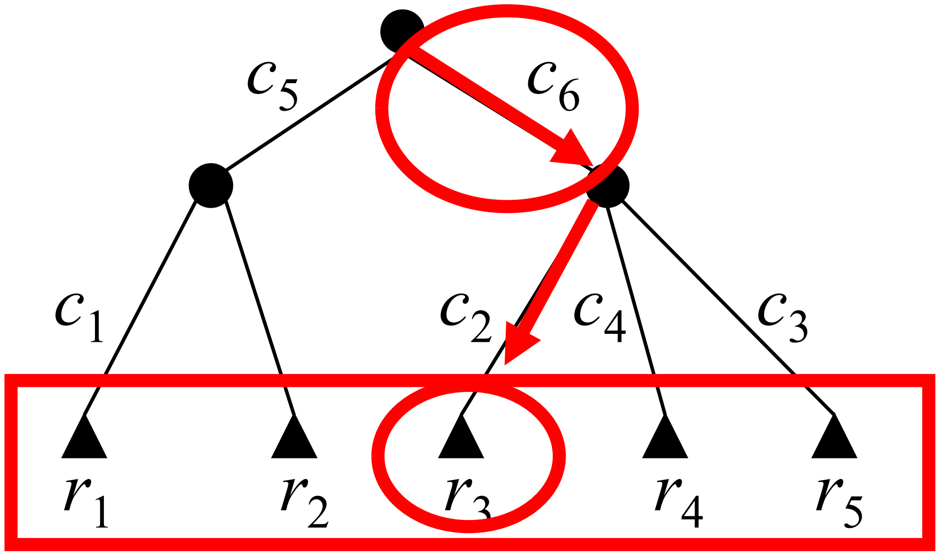
Perfect phylogenies

- A *perfect phylogeny* (PP) is
  a rooted tree T representing the evolutionary history of *m objects* in terms of *n characters*



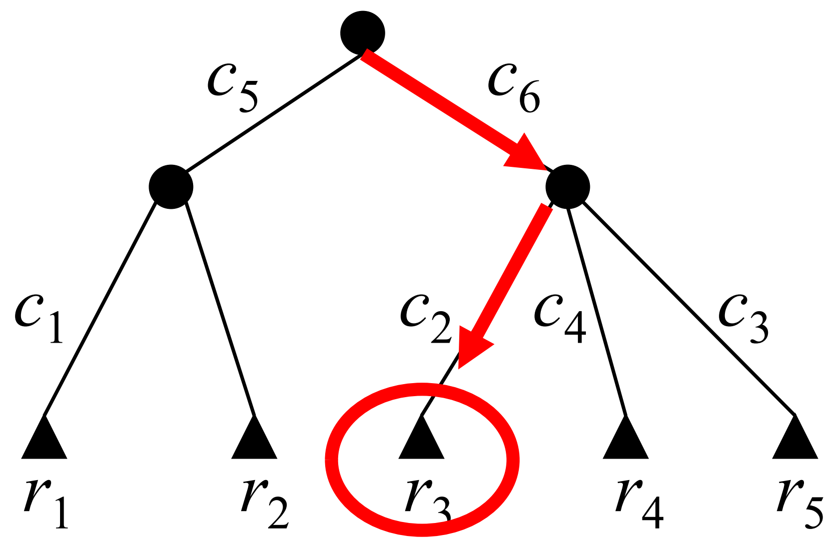
Perfect phylogenies

- The leaves of T correspond bijectively to the objects.
- Each character labels an edge of T
- An object r exhibits a character c if and only if c is on the path from the root to r



Perfect phylogenies

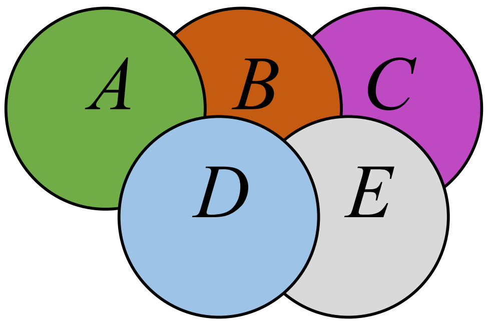
- The *matrix representation* of a PP is a binary matrix M such that:
  1. each row is associated with an object
  2. each column is associated with a character
  3. for a row r and a column c, $M_{r,c} = 1$ if and only if r exhibits c



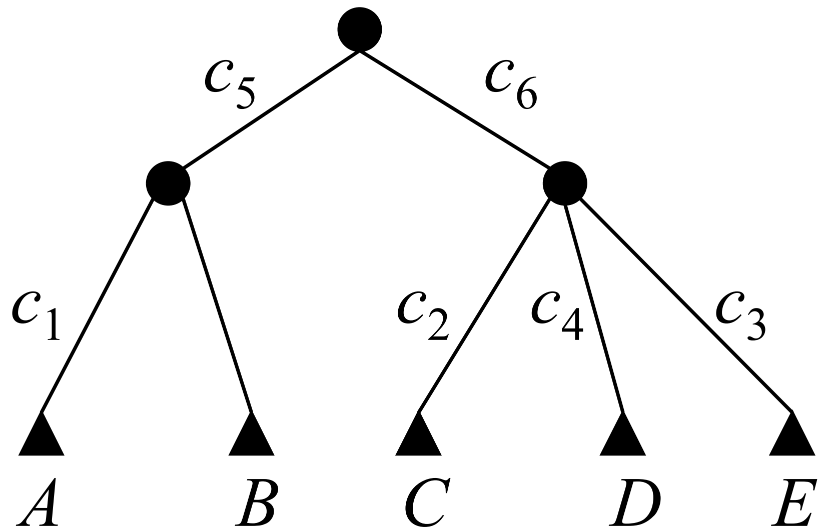
M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1				1	
r_2					1	
r_3		1				1
r_4				1		1
r_5			1			1

Motivation

- Tumor progression is assumed to admit a perfect phylogeny [7], where:
  - each object is a tumor subclone
  - each character is a somatic mutation
- This phylogenetic tree can offer a more comprehensive knowledge of tumor progression [2, 11]



Tumor subclones

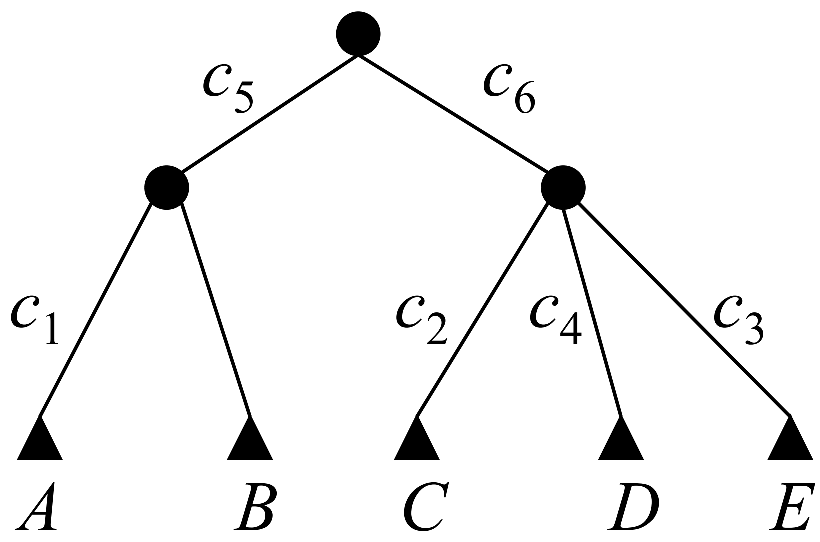


Motivation

- DNA sequencing technologies can help us to obtain the matrix representation and thereby reconstruct the PP

M	c_1	c_2	c_3	c_4	c_5	c_6
A	1				1	
B					1	
C		1				1
D				1		1
E			1			1

sequencing data

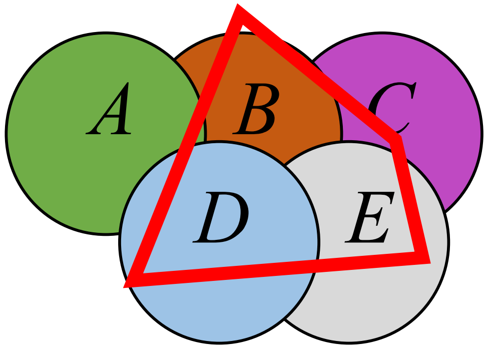


reconstructed PP

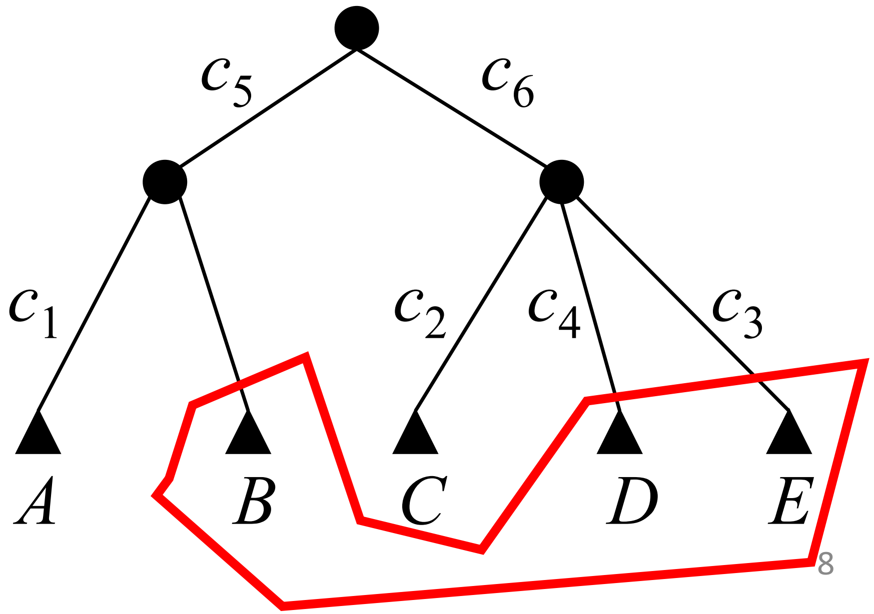
Motivation

- However, most data used currently are obtained from *bulk sequencing* due to cost effectivity

- In such data, each tumor sample may contain more than one subclones



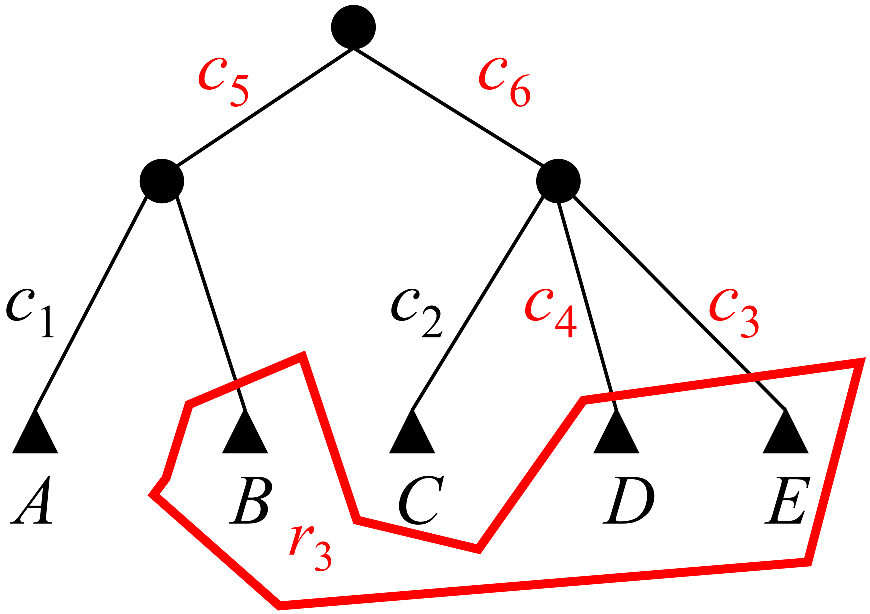
a sample



Motivation

- In this case, a mutation is observed in a sample if it is carried by any of the subclones mixed in the sample

- The data obtained by bulk sequencing may not exhibit a perfect phylogeny



	c_1	c_2	c_3	c_4	c_5	c_6
E			1			1
B					1	
D				1		1

r_3			1	1	1	1

Motivation

- To reconstruct the PP from bulk sequencing data, two optimization problems were proposed

- We begin by the *minimum split-row problem* (MSRP) [7]

- **Note**: all matrices in this slides are binary

Definition

- Given a matrix M, a *split-row operation* on M split a row r of M into several rows whose bitwise OR is r

- In MSRP, each split-row operation is associated with a cost: the number of additional rows

r_3			1	1	1	1

cost = 2

E			1			1
B					1	
D				1		1

Definition

- The goal is to perform split-row operations such that:
  1. The resulting matrix corresponds to a PP
  2. The total cost is minimum possible

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1		1	1	1
r_2					1	
r_3			1	1	1	1
r_4		1				1

cost = 4

M'	c_1	c_2	c_3	c_4	c_5	c_6
$r_1^{(1)}$	1				1	
$r_1^{(2)}$		1				1
$r_1^{(3)}$				1		1
$r_2^{(1)}$					1	
$r_3^{(1)}$			1			1
$r_3^{(2)}$					1	
$r_3^{(3)}$				1		1
$r_4^{(1)}$		1				1

Notation

- We need the following

- m: the number of rows of M
- n: the number of columns of M
- $\varepsilon(M)$: minimum cost of transforming M into a matrix corresponding to a PP

- O^*-notation: suppress the factors polynomial in input size
- Example: $mn^2 2^n = O^*(2^n)$

Definition

- Parameterized MSRP:

  Input: a matrix M, an integer $k > 0$

  Output: is $\varepsilon(M) \leq k$?

  Parameter: k

- A problem is *fixed-parameter tractable* (FPT) [4] with respect to a parameter k if it can be solved in $f(k) \cdot (|I| + k)^{O(1)}$ time, where I is the input instance

- Example: $O(2^k n^3) = O^*(2^k)$ time

Related work

- MSRP was proposed by Hajirasouliha and Raphael [7]
- Hujdurović *et al*. [9] showed that MSRP is NP-hard and gave an efficient heuristic algorithm
- Later, Hujdurović *et al*. [8] proved the APX-hardness of MSRP and gave exact and approximation algorithms

Related work

- Husić *et al*. [10] formulated MSRP as an *Integer Linear Program* (ILP)

- The ILP is implemented in the software package MIPUP [10]

- Sheu *et al*. [12] showed that MSRP is fixed-parameter tractable

Related work

Source	Complexity
naive [8]	$2^{\Omega(nm)}$
[8]	$O^*(n^n)$
[10]	- (ILP)
[12]	$O^*(2^{\min(n,\, 2\varepsilon(M))})$ ← **FPT-time**

Exact algorithms for MSRP (APX-hard)

Definition

- We study the *kernelization* of MSRP

- Kernelization is a mathematical concept that aims to analyze preprocessing algorithms.

- It is defined as follows

- Two instances (I, k) and (I', k') are *equivalent* if
  (I, k) is a yes-instance $\leftrightarrow$ (I', k') is a yes-instance

Definition

- A *kernelization algorithm*, or a *kernel*, is an algorithm that
  given an instance (I, k),
  works in time polynomial in $|I| + k$,
  returns an equivalent instance (I', k') such that
  $|I'| + k'$ is bounded by a computable function of k

- The output of a kernelization algorithm is also called a *kernel*

- Example: $m \times n$ matrix $\rightarrow$ $3k \times 4k$ matrix (kernel)

Contribution

- It is known that a problem is FPT if and only if it admits a *kernel* [4]

- Thus, Sheu *et al.*'s FPT result implies that
  MSRP admits a kernel (of exponential size)

- We show that MSRP admits a kernel with
  the numbers of rows and columns both linear to $\varepsilon(M)$

Contribution

Kernelization algorithms for MSRP

Source	Size	Time
implied by [12]	exponential in $\varepsilon(M)$	polynomial
[this]	$3\varepsilon(M)$ rows $4\varepsilon(M)$ columns	$O(mn^{1.373} + n^3)$

m: number of rows of M
n: number of columns of M

Related work

- Hujdurović *et al.*'s [8] approximation algorithms are based on an equivalent formulation, called the *branching formulation*, of MSRP

- In this formulation, the input matrix M is represented by a directed acyclic graph D_M

- The ratio of their approximation algorithms are measured by, respectively, the *height* and *width* of D_M

- The precise definitions will be given later

Related work (wrong)

- Let $h(M)$ be the height of D_M
- Let $w(M)$ be the width of D_M

- Their result is summarized in the following table

Source	ratio	Complexity
[8]	$h(M)$	$O(mn^2)$
[8]	$w(M)$	$O(mn^2 + n^{3.373})$

Existing approximation algorithms for MSRP

Related work (correction)

- Let $h(M)$ be the height of D_M
- Let $w(M)$ be the width of D_M

- Their result is summarized in the following table

Source	Guarantee	Complexity
[8]	$h(M) \cdot (m + \varepsilon(M))$	$O(mn^2)$
[8]	$w(M) \cdot (m + \varepsilon(M))$	$O(mn^2 + n^{3.373})$

Existing approximation algorithms for MSRP

Contribution

- We give a new approximation algorithm with ratio
  $$2 \min(\lg n, \lg 2\varepsilon(M))$$

- Our algorithm improves on [8]'s algorithms:
- Given any matrix M, it finds a solution which is at least as good (in terms of cost) as the output of each of [8]'s algorithms

- In addition, it is faster than [8]'s $w(M)$-approximation algorithm

Contribution

Approximation algorithms for MSRP

Source	Guarantee	Complexity
[8]	$h(M) \cdot (m + \varepsilon(M))$	$O(mn^2)$
[8]	$w(M) \cdot (m + \varepsilon(M))$	$O(mn^2 + n^{3.373})$
[this]	$2\lg(n) \cdot \varepsilon(M)$	$O(mn^2 + n^3)$

always finds a solution not worse than
the outputs of [8]'s algorithms

Definition

- A variant of MSRP, called the *minimum distinct row split problem* (MDCRSP) [7], is also considered
- The only difference is that MDCRSP seeks to minimize the number of distinct rows

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1		1	1	1
r_2					1	
r_3			1	1	1	1
r_4		1				1

M'	c_1	c_2	c_3	c_4	c_5	c_6
$r_1^{(1)}$	1				1	
$r_1^{(2)}$		1				1
$r_1^{(3)}$				1		1
$r_2^{(1)}$					1	
$r_3^{(1)}$			1			1
$r_3^{(2)}$					1	
$r_3^{(3)}$				1		1
$r_4^{(1)}$		1				1

cost = 5 (distinct rows)

Related Work

- MDCRSP is APX-complete [8]
- Most exact algorithms for MSRP can be generalized to solve MDCRSP

Reference	Complexity
[8]	$O^*(n^n)$
[10]	- (ILP)
[12]	$O^*(2^{\min(n,\ 3\varepsilon(M))})$

Exact algorithms for MDCRSP

m: number of rows of M
n: number of columns of M

Contribution

- We give new approximation algorithms with improved ratios for MDCRSP

Source	Approximation ratio	Time
[8]	2	$O(mn^2)$
[this]	$5/3 \approx 1.67$	$O(mn^2)$
[this]	$\cancel{1.4 + \delta}$ for any $\delta > 0$	$n^{O(1/\delta)} \approx n^{64/\delta}$

$4/3 + \delta \approx 1.33 + \delta$

Approximation algorithms for MDCRSP

m: number of rows of M

n: number of columns of M

Outline

- Introduction
- **Preliminaries**
- A kernelization algorithm for MSRP
- Conclusion and future work

Review of [8]'s formulation

- Hujdurović *et al*. [8] proposed a new formulation, called the *branching formulation*, of MSRP

- They showed that MSRP is equivalent to finding an optimal spanning forest of a derived DAG

- The formulation is reviewed as follows

Definition

- For a column c of M, the *support* of c, denoted by $supp_M(c)$, is the set of rows r such that $M_{r,c} = 1$

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1		1	1	1
r_2					1	
r_3			1	1	1	1
r_4		1				1

$supp_M(c_6) = \{r_1, r_3, r_4\}$

Definition

- Consider two columns c and c'

- c and c' are *disjoint* if their supports are disjoint

M'	c_1	c_2	c_3	c_4	c_5	c_6
$r_1^{(1)}$	1				1	
$r_1^{(2)}$		1				1
$r_1^{(3)}$				1		1
$r_2^{(1)}$					1	
$r_3^{(1)}$			1			1
$r_3^{(2)}$					1	
$r_3^{(3)}$				1		1
$r_4^{(1)}$		1				1

Definition

- c *contains* c' if $supp_M(c) \supset supp_M(c')$.
- c and c' are *nested* if c contains c' or c' contains c

M'	c_1	c_2	c_3	c_4	c_5	c_6
$r_1^{(1)}$	1				1	
$r_1^{(2)}$		1				1
$r_1^{(3)}$				1		1
$r_2^{(1)}$					1	
$r_3^{(1)}$			1			1
$r_3^{(2)}$					1	
$r_3^{(3)}$				1		1
$r_4^{(1)}$		1				1

Definition

- c, c' are *in conflict* if:
  1. they are not disjoint
  2. they are not nested

- In other words, they have partial intersection

conflict

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1		1	1	1
r_2					1	
r_3			1	1	1	1
r_4		1				1

Assumptions

- For simplicity, we assume the input matrix contains no empty columns
- Removal of such columns does not change $\varepsilon(M)$

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1		1	1	1	1	1
r_2					1	
r_3				1	1	1
r_4		1	1			1

Definition

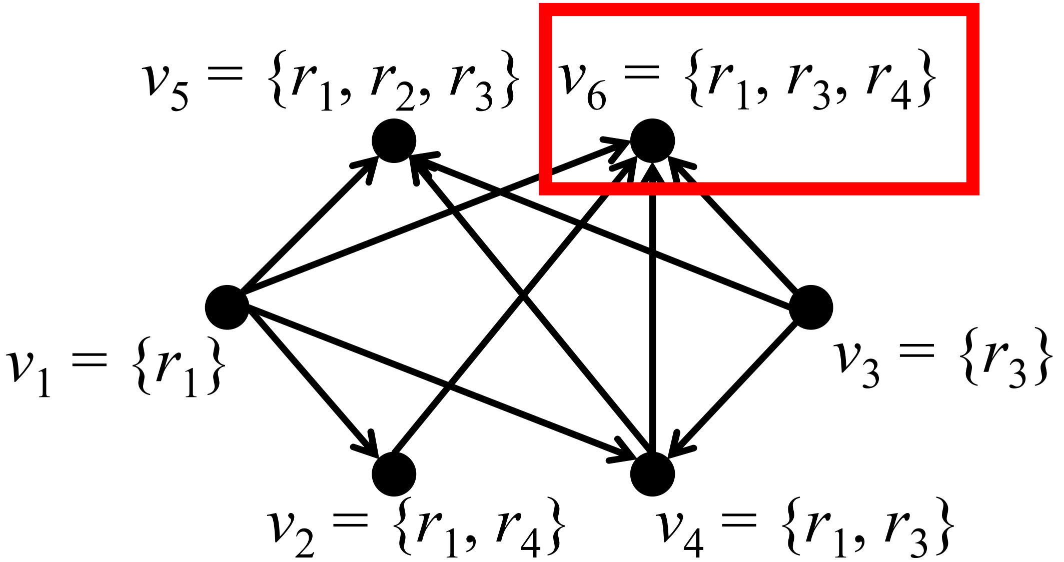
- Under the assumption, the relation between two columns c, c' is exactly one of:
    1. disjoint
    2. nested
    3. in conflict

- We say c, c' are *compatible* if they are not in conflict

Definition

- The *containment digraph* D_M [8] is a DAG such that:
    1. the vertex set is the set of (distinct) supports
    2. (v, v') is an arc of D_M iff $v \subset v'$

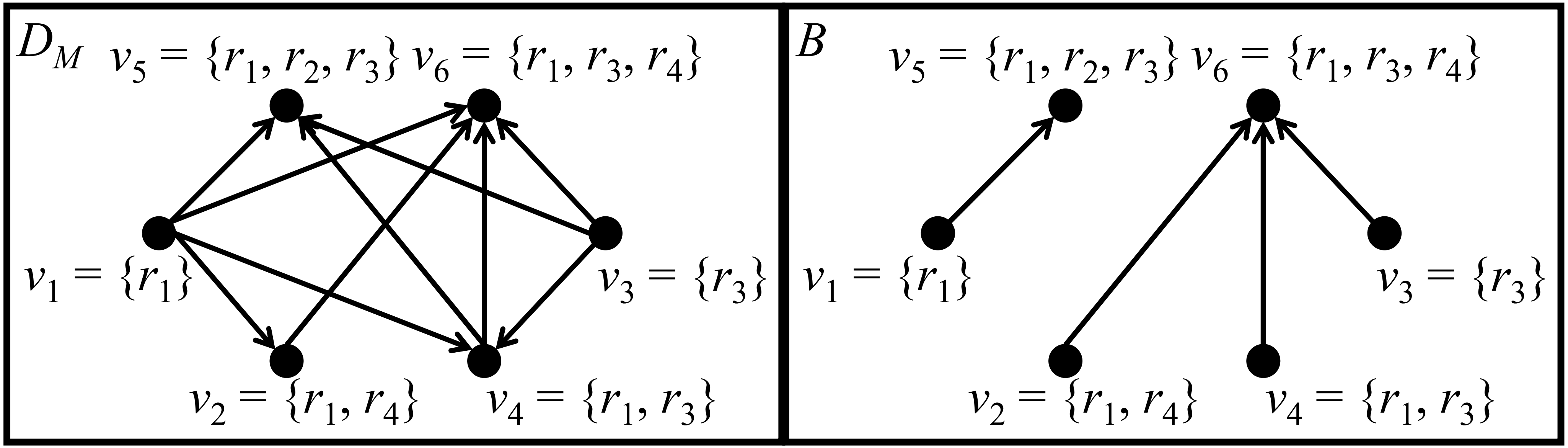
M	c_1	c_2	c_3	c_4	c_5	c_6	c_7
r_1	1	1		1	1	1	1
r_2					1		
r_3			1	1	1	1	1
r_4		1				1	1



Note: duplicate columns map to the same vertex

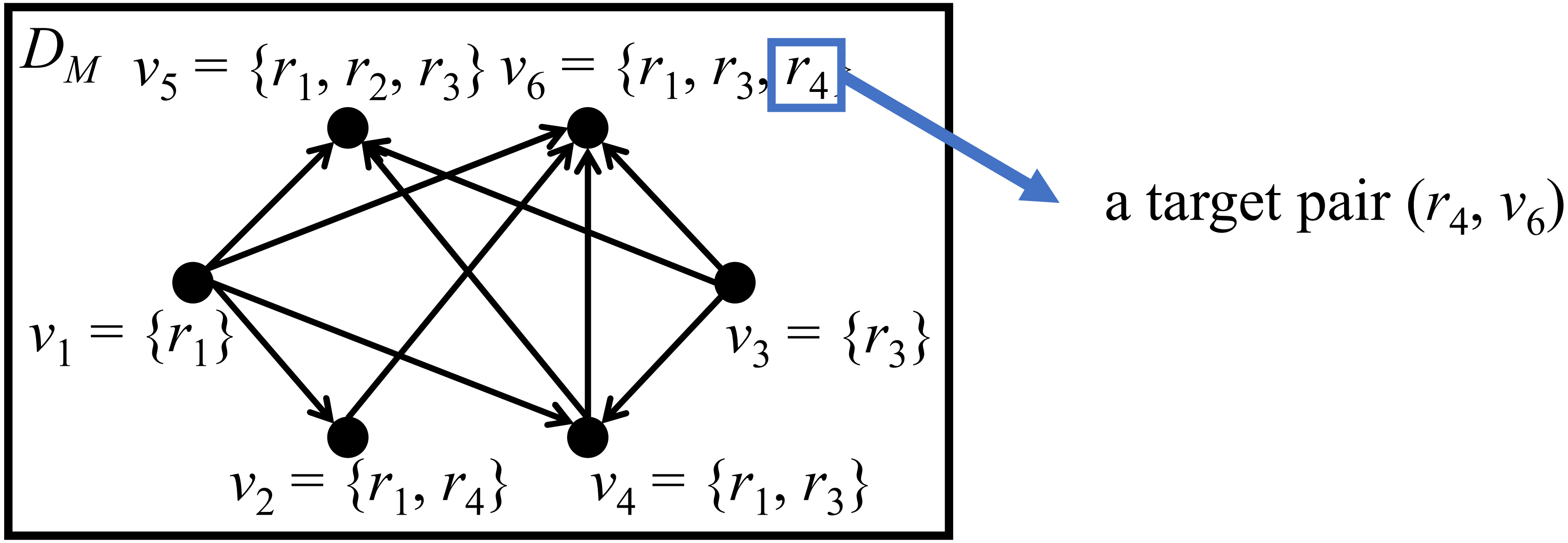
Definition

- A *branching* is a subset of arcs of D_M such that for each vertex v there is has at most one arc leaving v



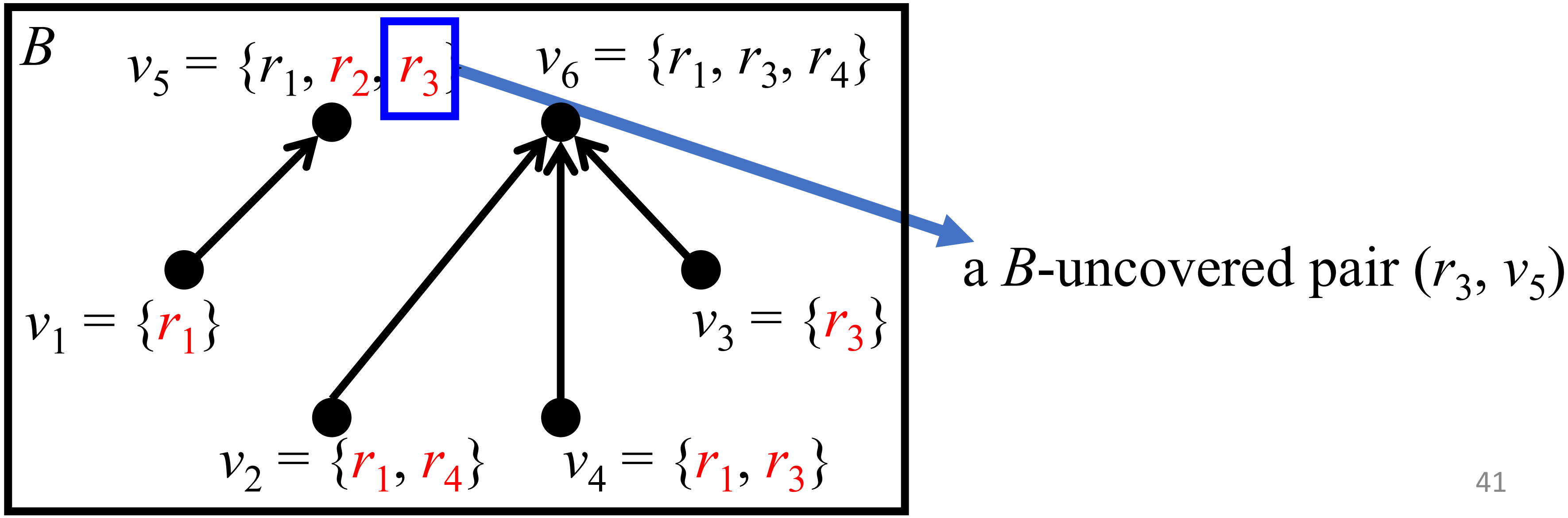
Definition

- For a row r and a vertex v, if $r \in v$, then (r, v) is a *target pair*
- In the branching formulation, each target pair (r, v) specifies a "demand":
  - vertex v demands a row r in its children



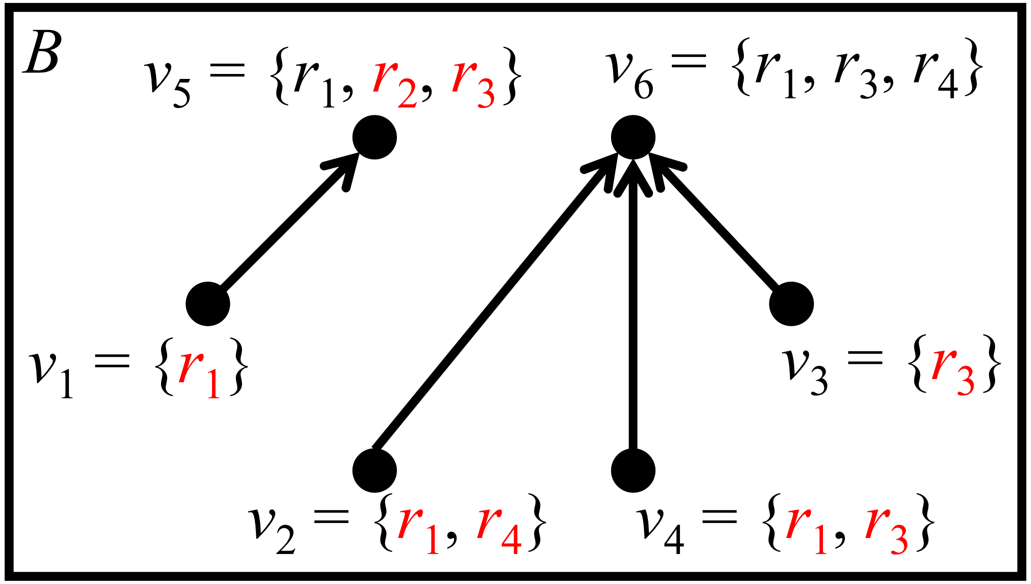
Definition

- A *B-uncovered pair* is a target pair (r, v) such that r is not in any child of v



Definition

- Let $U(B)$ be the set of all B-uncovered pairs
- The *cost* of B is defined as $|U(B)|$
- Solving MSRP is equivalent to finding the minimum cost branching [8]



cost = 8

The branching formulation

- Let $\beta(M)$ be the minimum cost of a branching

- **Theorem 2.1.** [8] $\beta(M) = m + \varepsilon(M)$

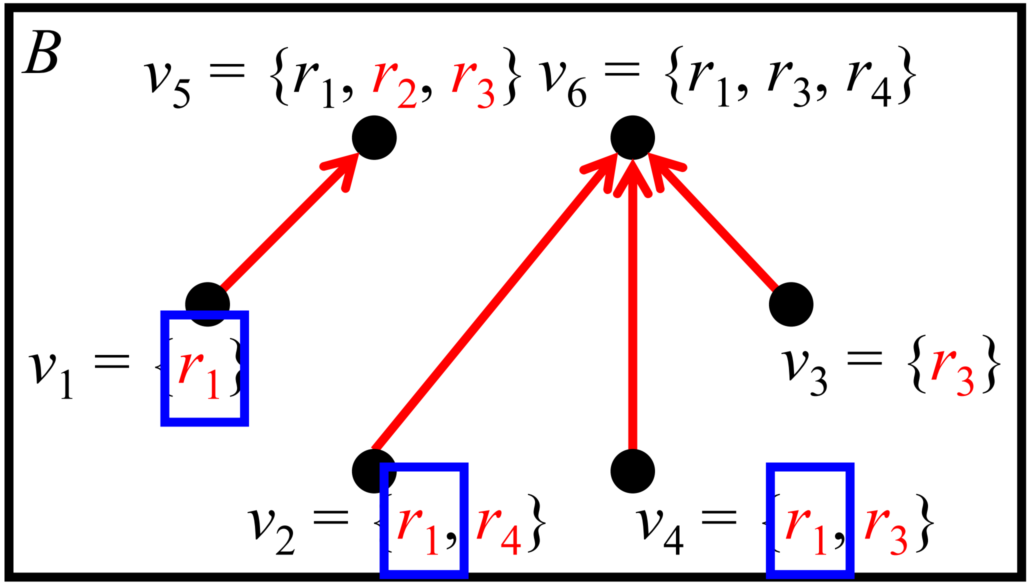
  m: the number of rows

- By Theorem 2.1, the finding of $\varepsilon(M)$ can be done by finding the optimal branching

The branching formulation

- Let $U_B(r)$ be the set of uncovered pairs contributed by r

- Each row r contributes $|U_B(r)|$ to the cost of B



$U_B(r_1)$: $\{(r_1, v_1), (r_1, v_2), (r_1, v_4)\}$

The branching formulation

- Consider an optimal branching B^*
- By Theorem 2.1, $|U(B^*)| = m + \varepsilon(M)$

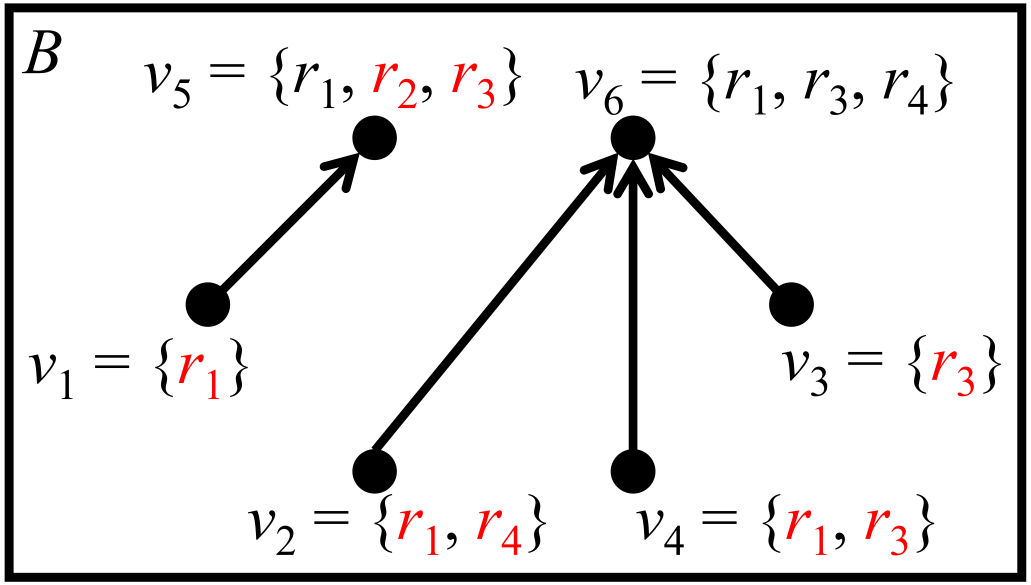
$$\rightarrow \Sigma_r\, |U_{B*}(r)| = m + \varepsilon(M)$$

$$\rightarrow \Sigma_r\, (|U_{B*}(r)| - 1) = \varepsilon(M)$$

- We say that *r contributes* $|U_{B*}(r)| - 1$ to $\varepsilon(M)$

Notation

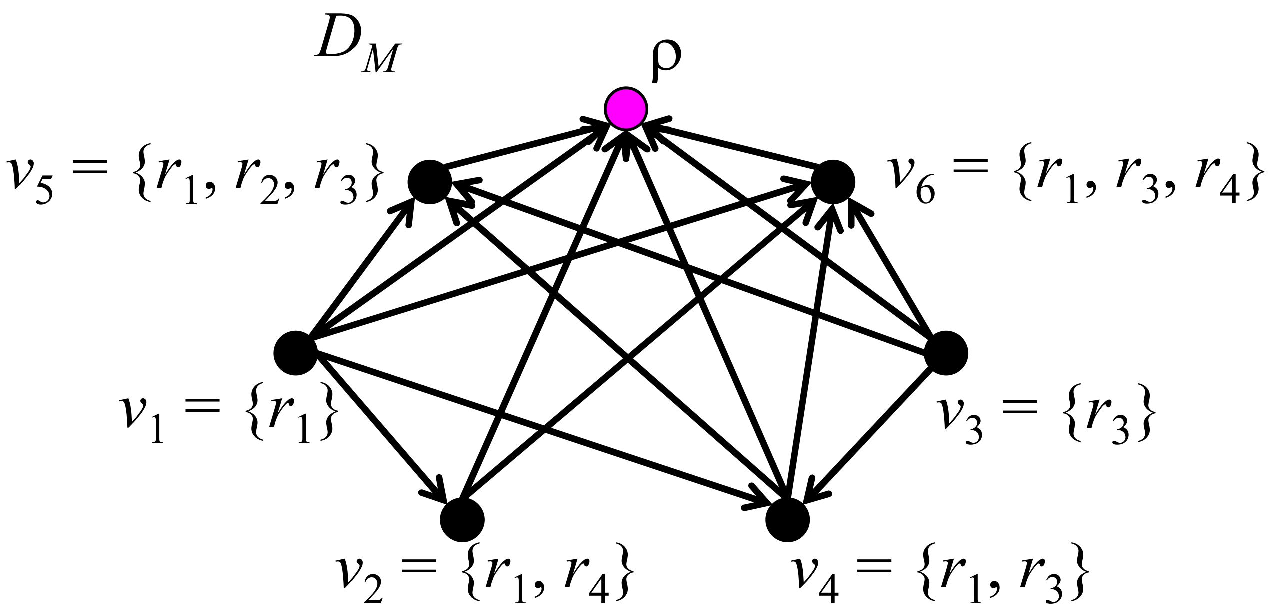
- *B-parent of v*: the parent of v in $(V(D_M), B)$
- *B-child of v*: a child of v in $(V(D_M), B)$
- $p_B(v)$: the B-parent of v



$p_B(v_4) = v_6$

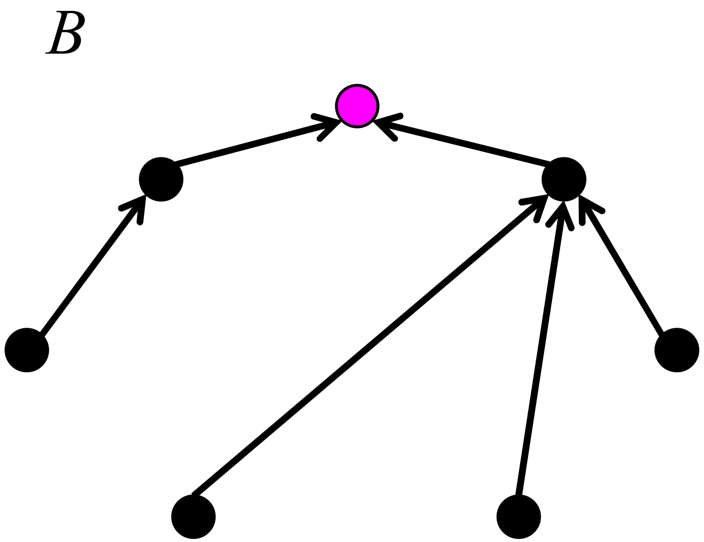
Assumption

- As in [12], we assume that M has a column whose entries are all ones.
- Denote the support of this column by ρ
- We call ρ *the root vertex*



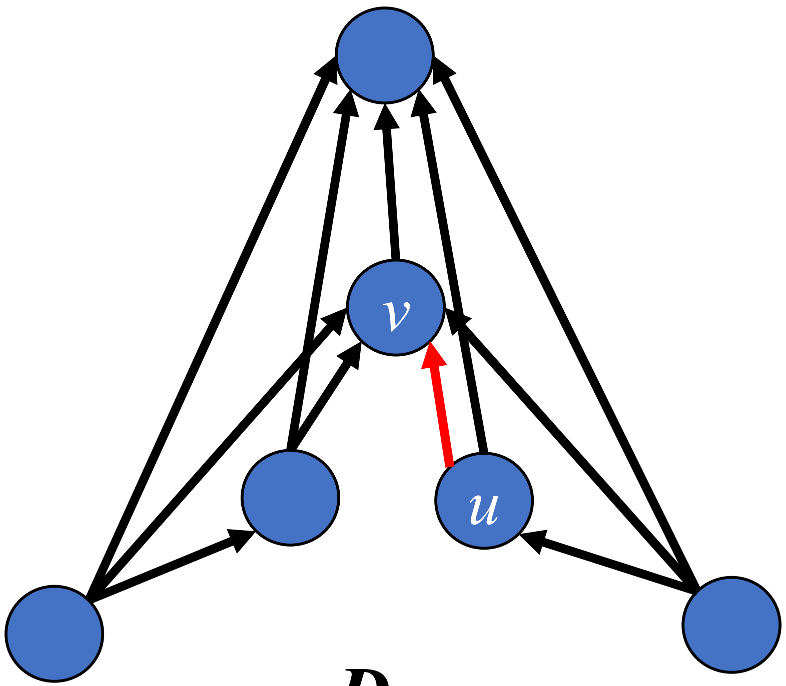
Assumption

- Thus, we may assume that D_M has an optimal branching with $(V(D_M), B)$ being a tree rooted at ρ

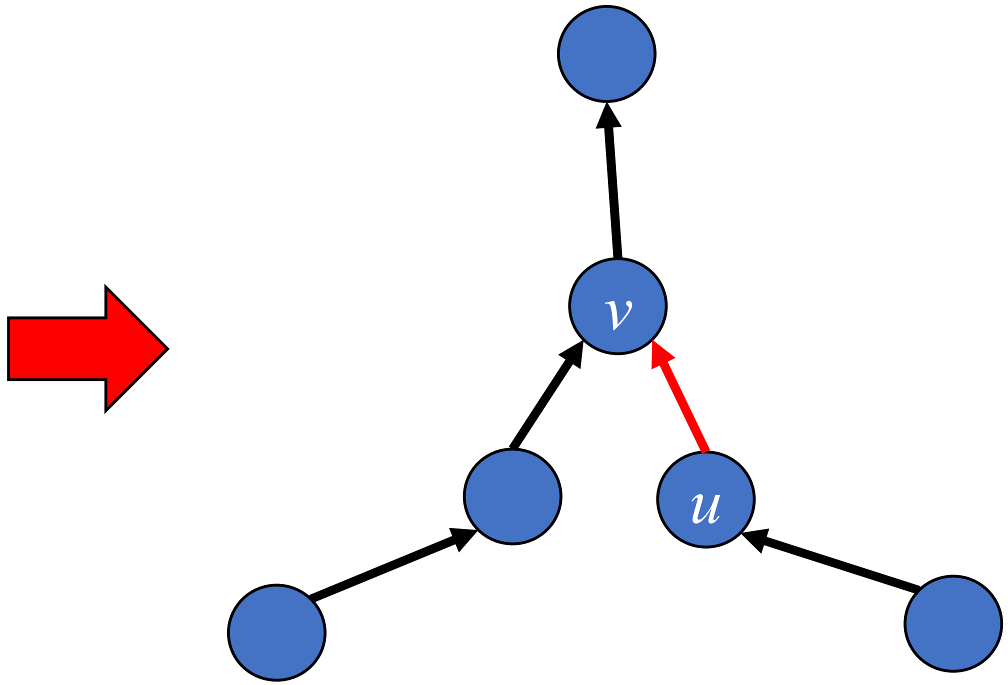


Notation

- For clarity, in our examples, we only display *elementary arcs* [1, 12] of D_M, where
- an arc (u, v) is elementary if there exists **no** w such that (u, w) and (w, v) are both arcs of D_M



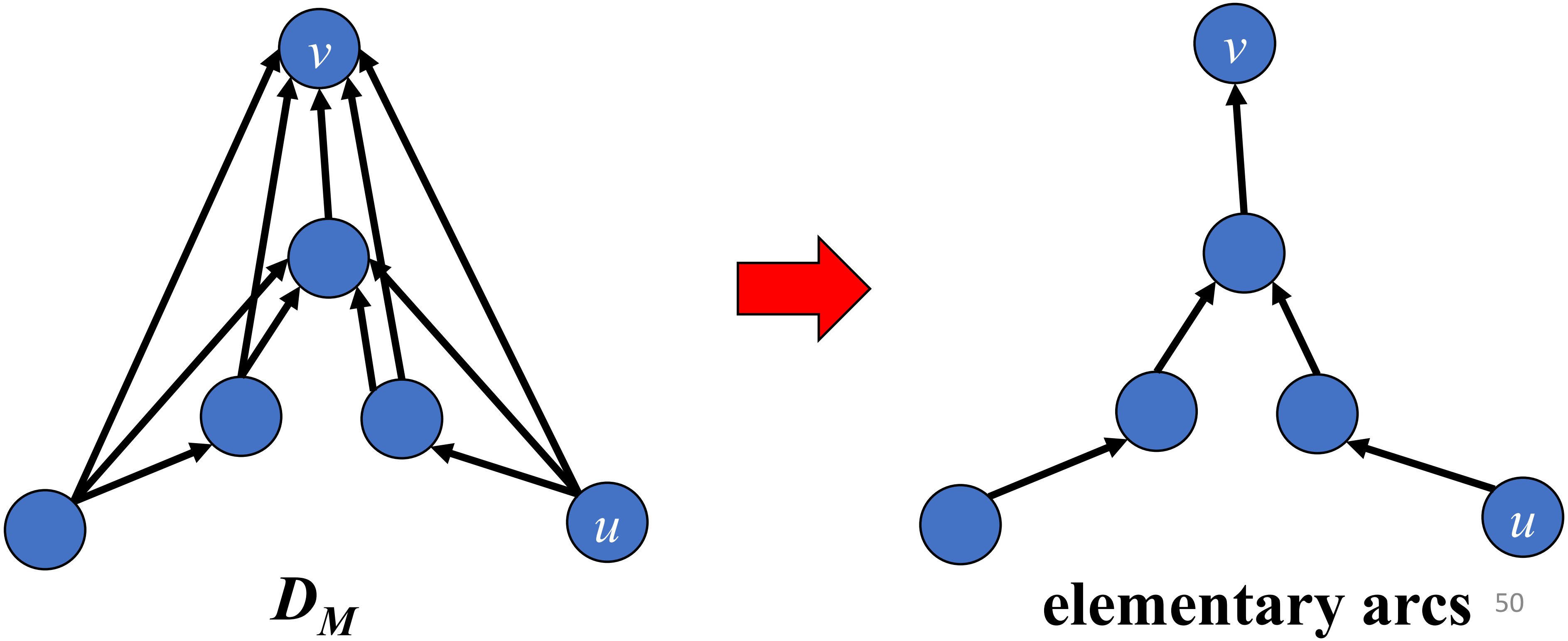
$\boldsymbol{D_M}$



elementary arcs

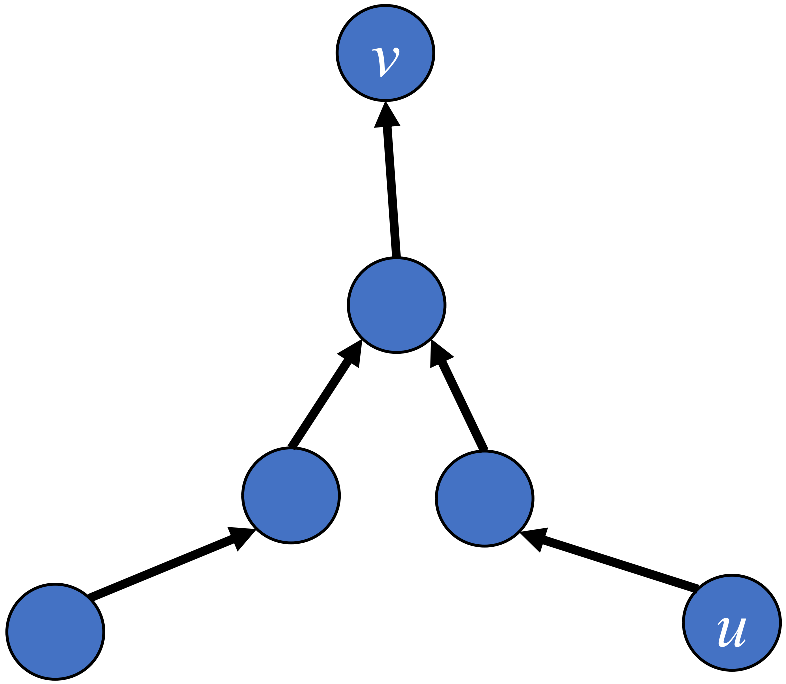
Notation

- The set of elementary arcs has the following property [1]:
  for any two vertices u, v, (u, v) is an arc of D_M
  if and only if
  there is a path consisting of only elementary arcs from u to v

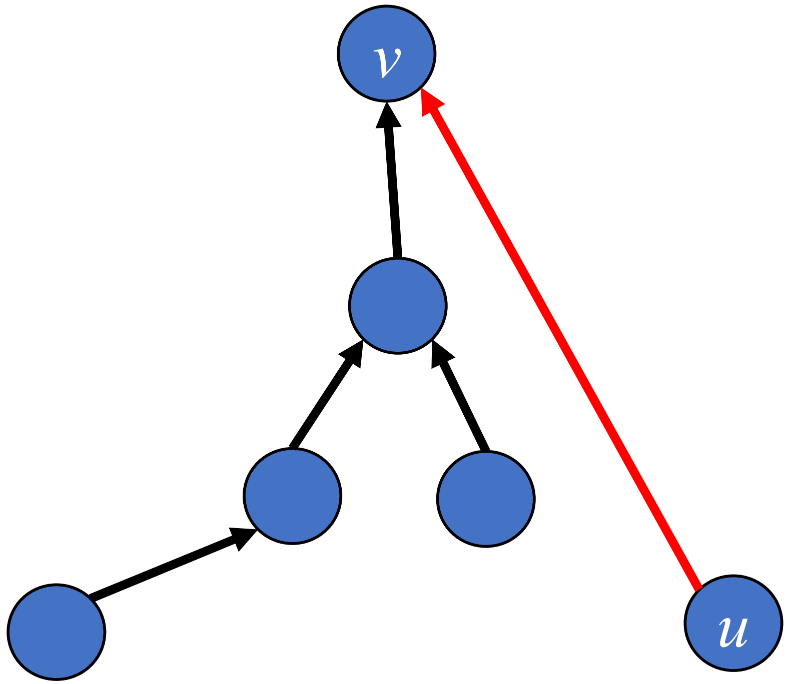


Remark

- We remark that omitting non-elementary arcs is simply for clarity of illustration
- A branching may contain non-elementary arcs



elementary arcs



a branching

Outline

- Introduction
- Preliminaries
- A kernelization algorithm for MSRP
    - Definition
    - Kernel size
    - Correctness
    - The algorithm
- Conclusion and future work

Idea

- Let M be a matrix

- The idea of our kernelization is to
  remove a subset of rows and columns to obtain a matrix
  M^- where $\varepsilon(M^-) = \varepsilon(M)$

Reduction rule

- **Rule 1.** If M contains a pair of duplicate columns c_i, c_j, remove one of them.

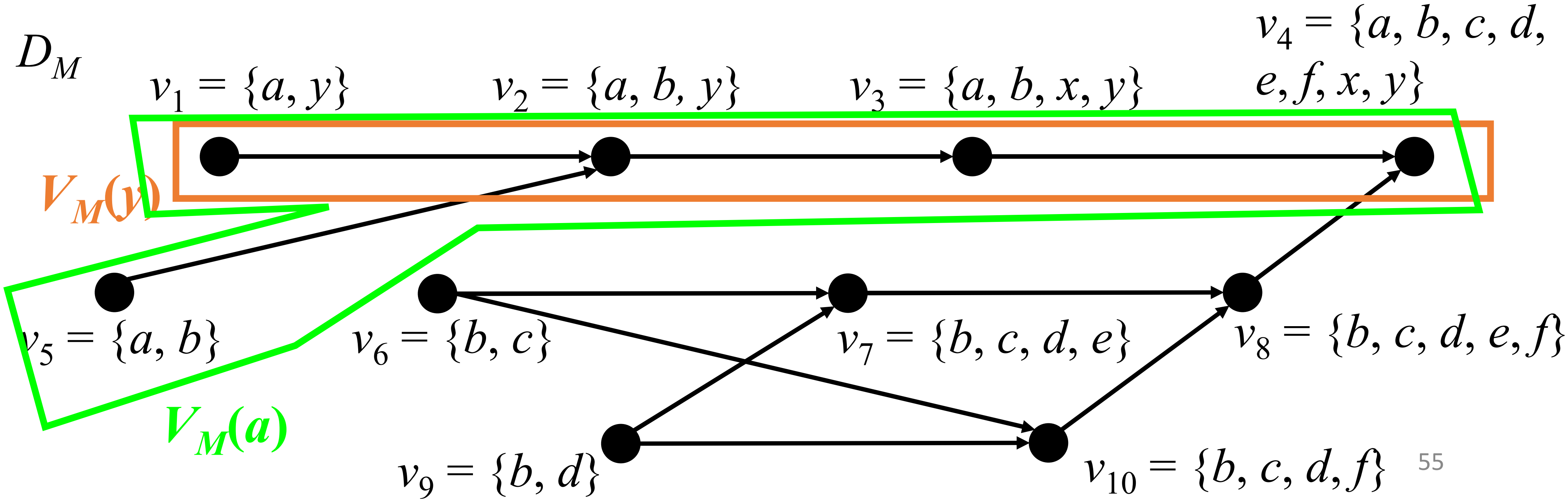
M	c_1	c_2	c_3	c_4	c_5	c_6	c_7
r_1	1	1		1	1	1	1
r_2					1		
r_3			1	1	1	1	1
r_4		1				1	1

$\Rightarrow$

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1		1	1	1
r_2					1	
r_3			1	1	1	1
r_4		1				1

Definition

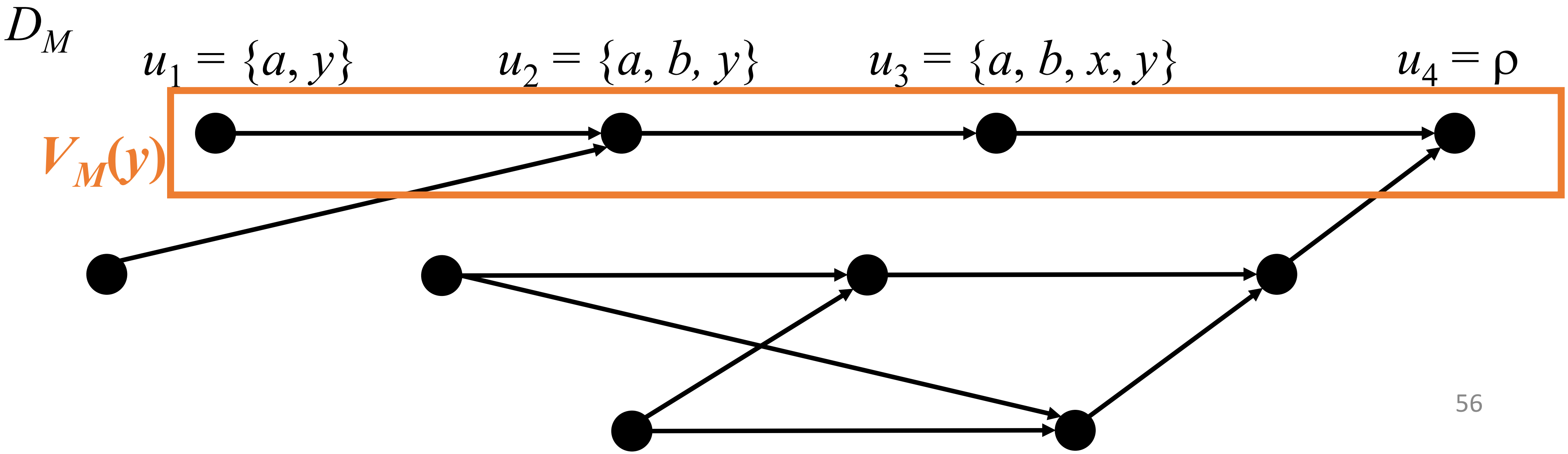
- For a row r of M, let $V_M(r)$ be the set of vertices containing r in $V(D_M)$.
- A row r is *chain-like* if any two vertices $u, v \in V_M(r)$ are nested.



Observation

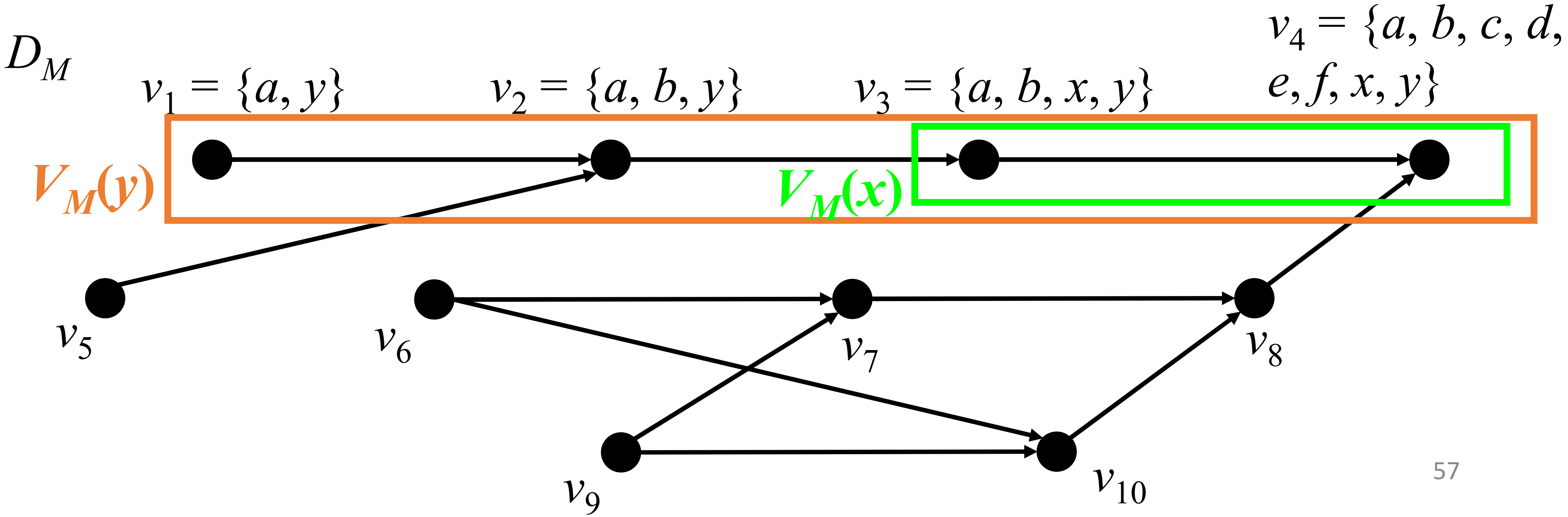
- Let r be a chain-like row. Order the vertices of $V_M(r)$ into $(u_1, u_2, \ldots, u_k)$ such that $|u_1| \leq |u_2| \leq \ldots \leq |u_k|$

- **Observation.** $u_1 \subset u_2 \subset \ldots \subset u_k$



Definition

- A row x is *doubly-chain-like* if:
  there exists a chain-like row y with $V(M, x) \subseteq V(M, y)$
- Note: every doubly-chain-like row is chain-like



Reduction rule (last time)

- **Rule 2 (last time).** If

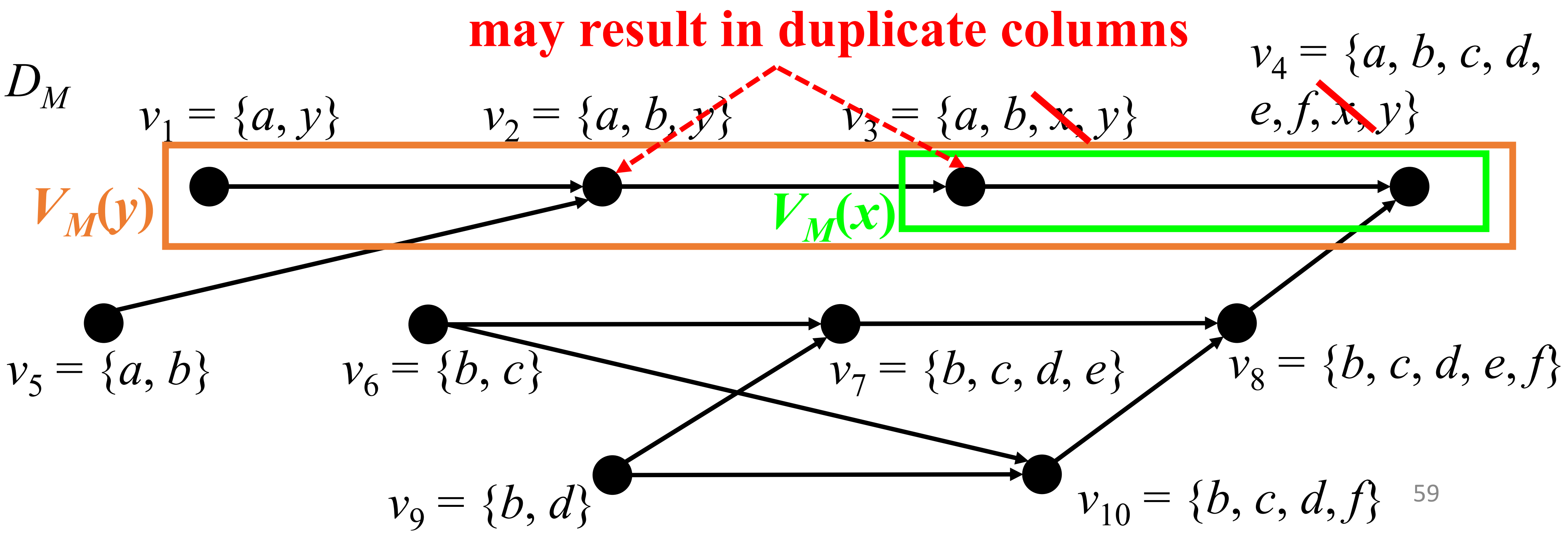
  (1) Rule 1 is not applicable to M, and

  (2) M has a doubly-chain-like row r,

  remove r.

Reduction rule

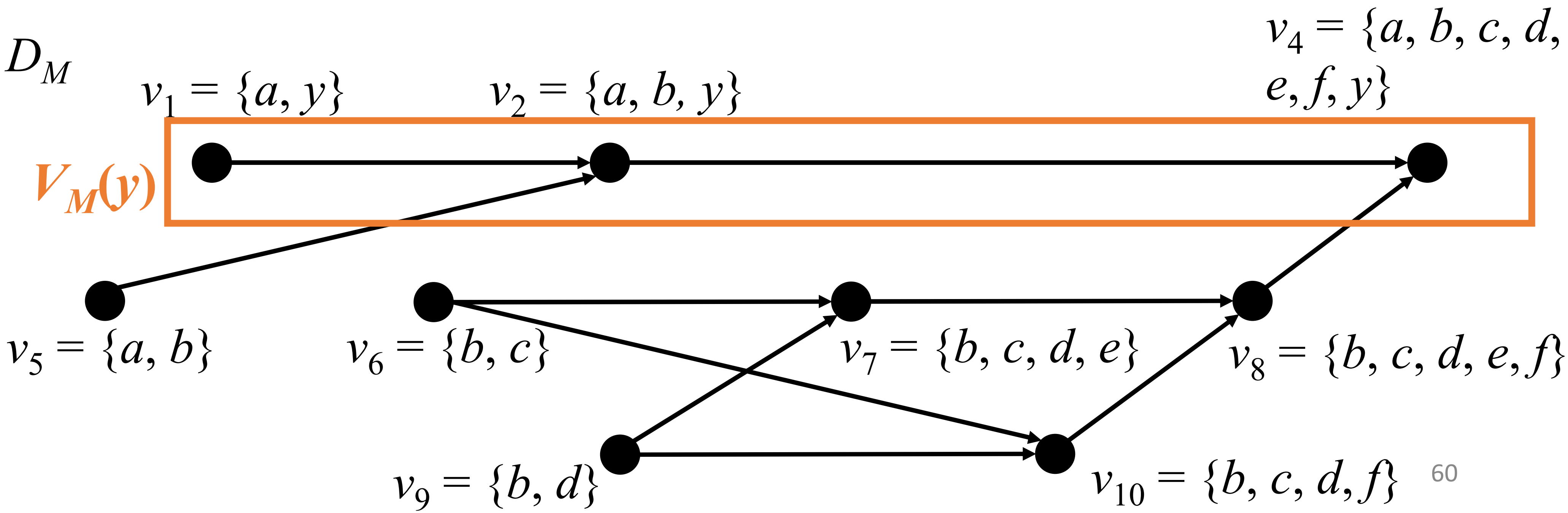
- **Rule 2.** If M has a doubly-chain-like row r, remove r



Reduction rule

- The containment digraph after performing Rule 2:

- The duplicate columns c_2, c_3 are both represented by v_2



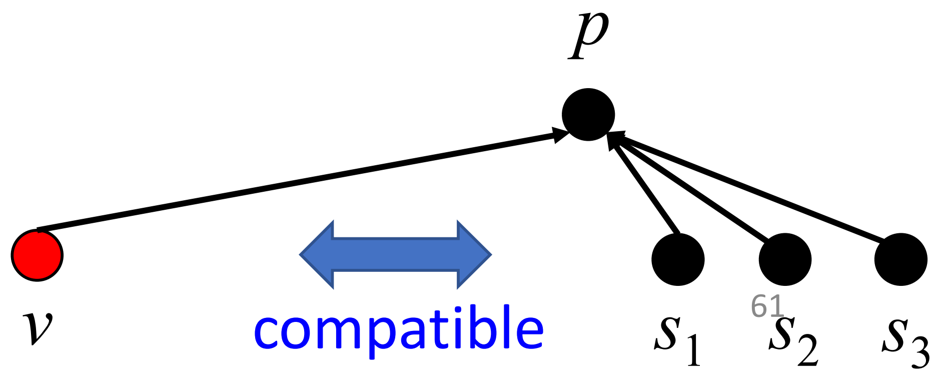
Definition

- A vertex v is *sibling-compatible* [12] if there is a vertex p:
  (1) $v \subset p$
  (2) for all vertices s such that $(s, p) \in A(D_M)$
  s and v are compatible.

$A(D_M)$: the set of arcs

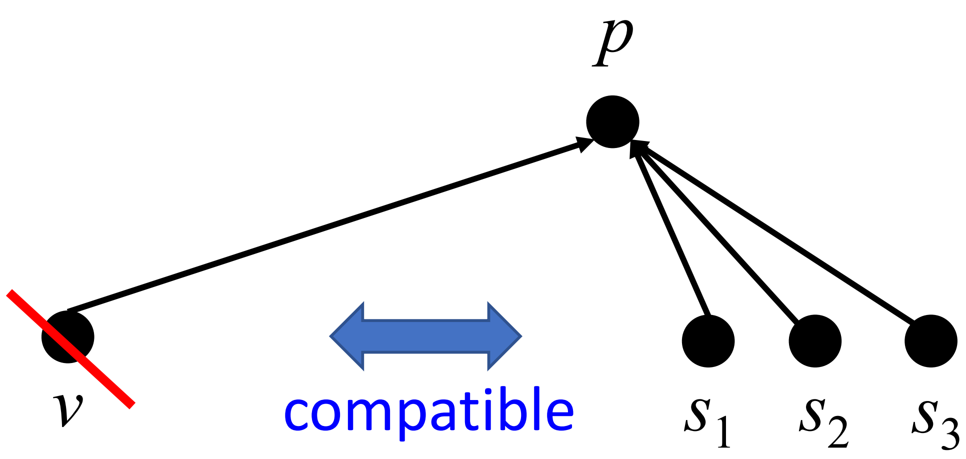
- Note: if v is the only vertex with $v \subset p$, (2) is vacuously true

- For any (v, p) such that the above holds, we say v is *sibling-compatible at* p



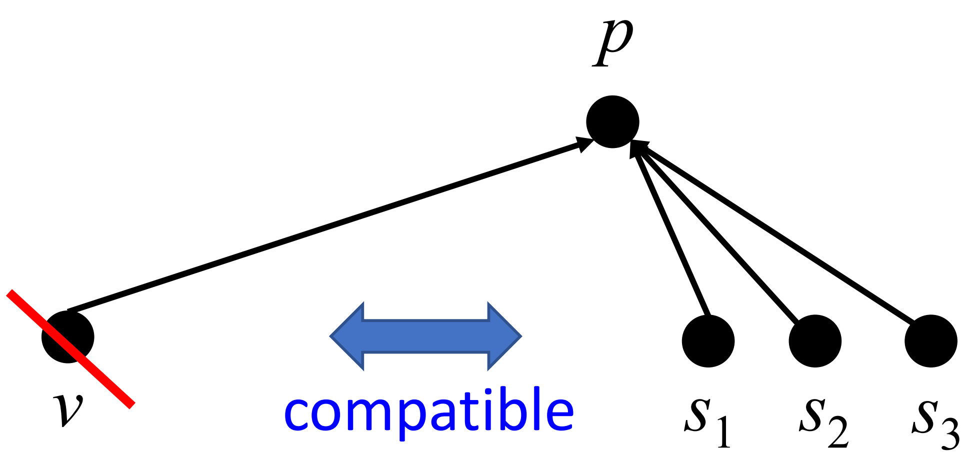
Reduction rule (last time)

- **Rule 3 (last time).** If
  (1) Rule 1 is not applicable to M, and
  (2) D_M contains a sibling-compatible vertex v,
  then remove the column c with $supp_M(c) = v$.



Reduction rule

- **Rule 3.** If D_M contains a sibling-compatible vertex v, then remove all columns c with $supp_M(c) = v$.



Kernelization algorithm

- A matrix is *reduced* if Rules 1, 2 and 3 are not applicable

- Our algorithm
  performs Rules 1, 2 and 3 on the input matrix exhaustively
  to obtain a reduced matrix

- To show that our algorithm is a kernelization, we analyze:
  - kernel size
  - safeness
  - time complexity

Outline

- A kernelization algorithm for MSRP
  - Definition
  - Kernel size
  - Safeness
  - The algorithm

Assumption

- Let M be a reduced matrix

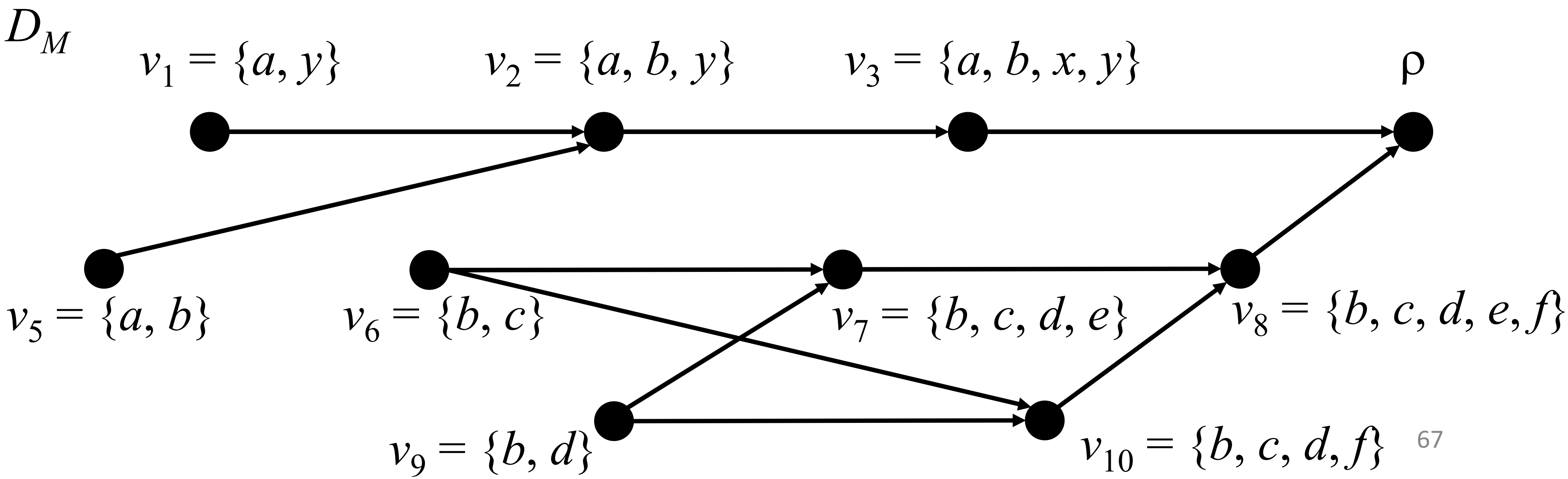
- We assume that $\varepsilon(M) > 0$
- Otherwise, the problem can be solved in polynomial time [6]

- In this section, we give upper bound on the size of M

- We first derive an upper bound on the number of rows

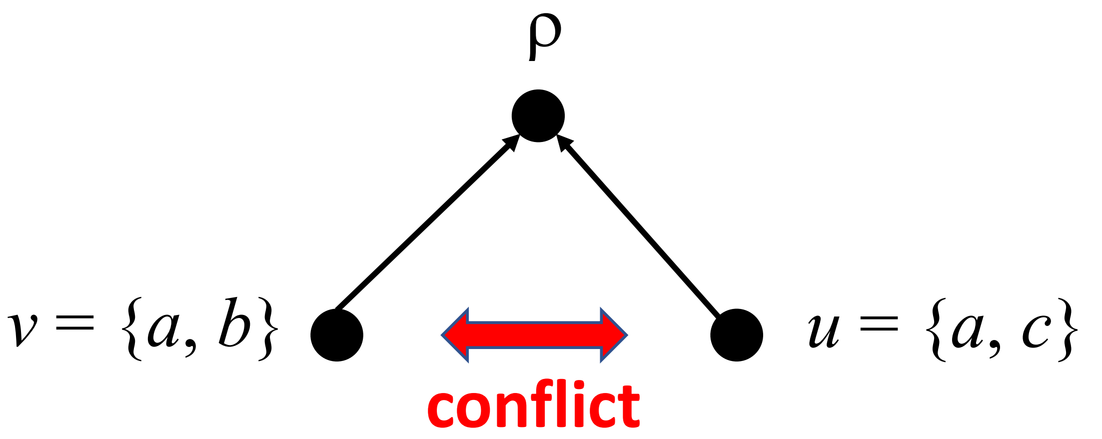
Lemma

- **Lemma 3.4.** Every vertex $v \in V(D_M)$ satisfies:
  (1) v contains at least two rows
  (2) v contains at least one non-chain-like rows



Proof

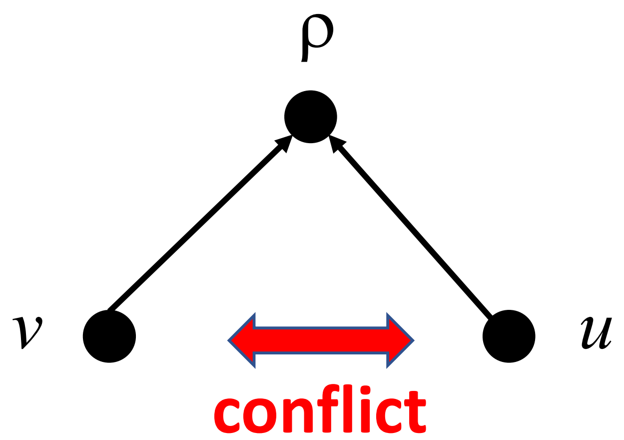
- Consider a non-root vertex v
- Note that $v \subset \rho$ and v is not sibling-compatible at ρ
- Thus, v is in conflict with some vertex $u \subset \rho$



Proof

- By the definition of conflict, we know that
  - $|v \cap u| \geq 1$ conflict $\leftrightarrow$ partial intersection
  - $|v - u| \geq 1$

- Since there is a row r in $v \cap u$ and r is non-chain-like:
  (1) v contains a non-chain-like row, and
  (2) $|v| > 1$



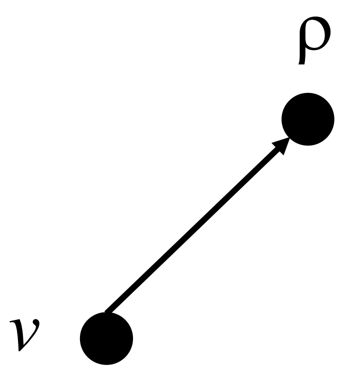
Proof

- Consider the root vertex ρ

- Since $\varepsilon(M) > 0$, D_M contains at least one non-root vertex v

- Since v satisfies (1) and (2) and $v \subset \rho$,
  ρ satisfies (1) and (2)

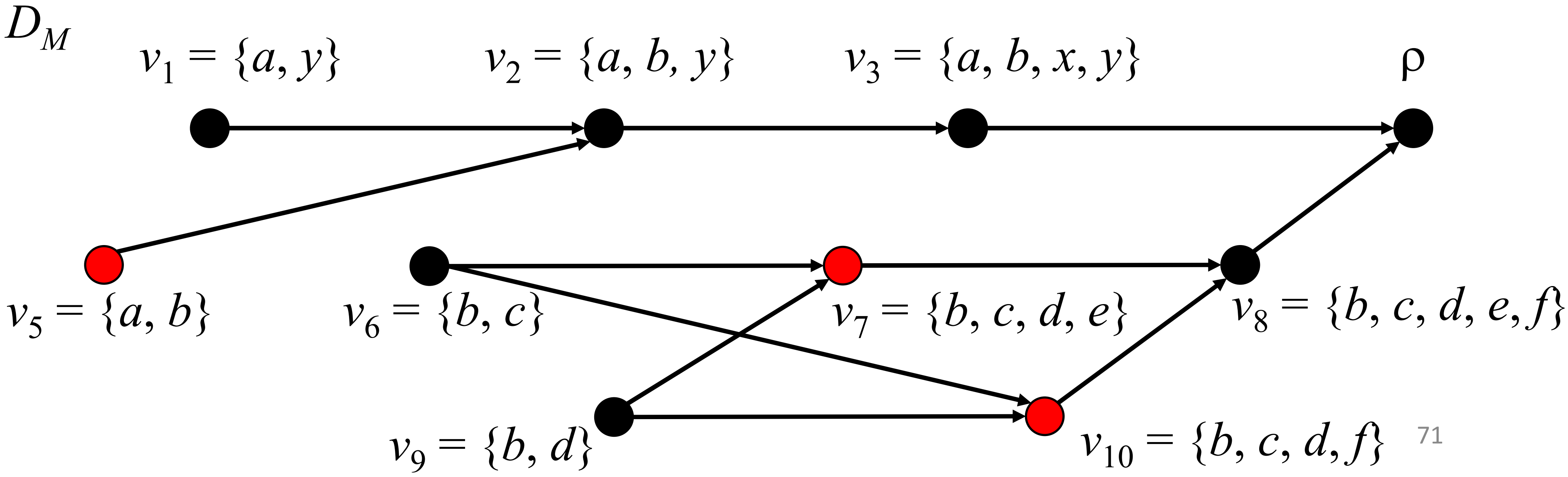
- This completes the proof



Definition

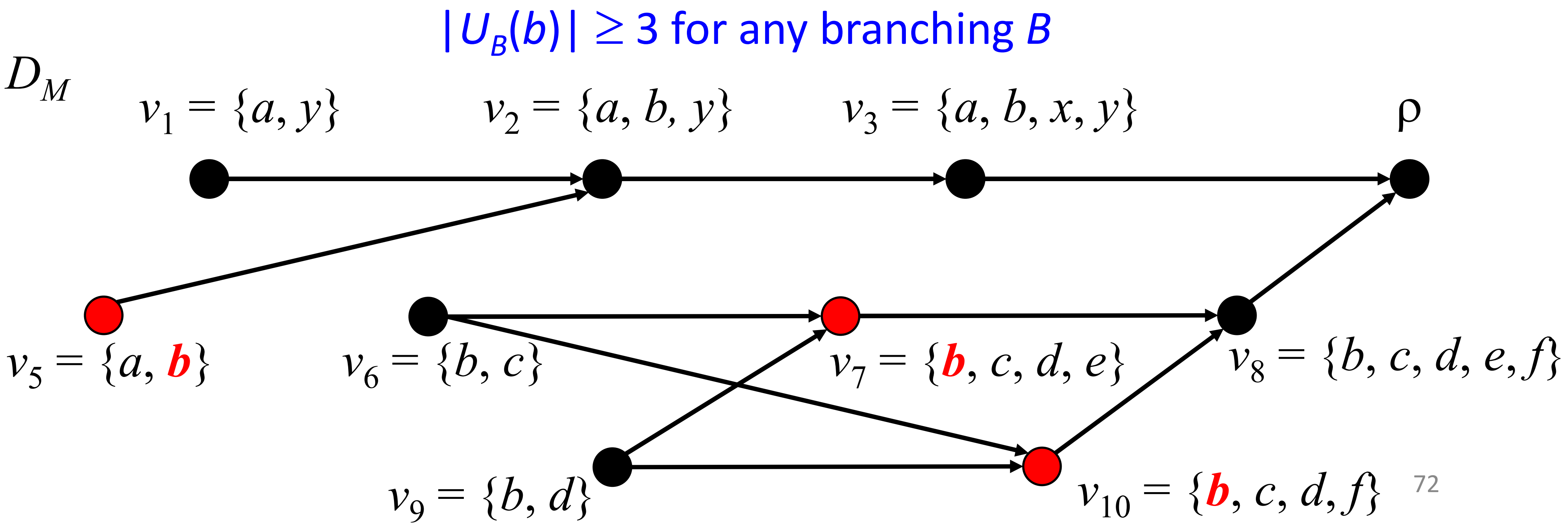
- An *antichain of D_M* is a set of pairwise non-nested vertices

- Example: $\{v_5, v_7, v_{10}\}$



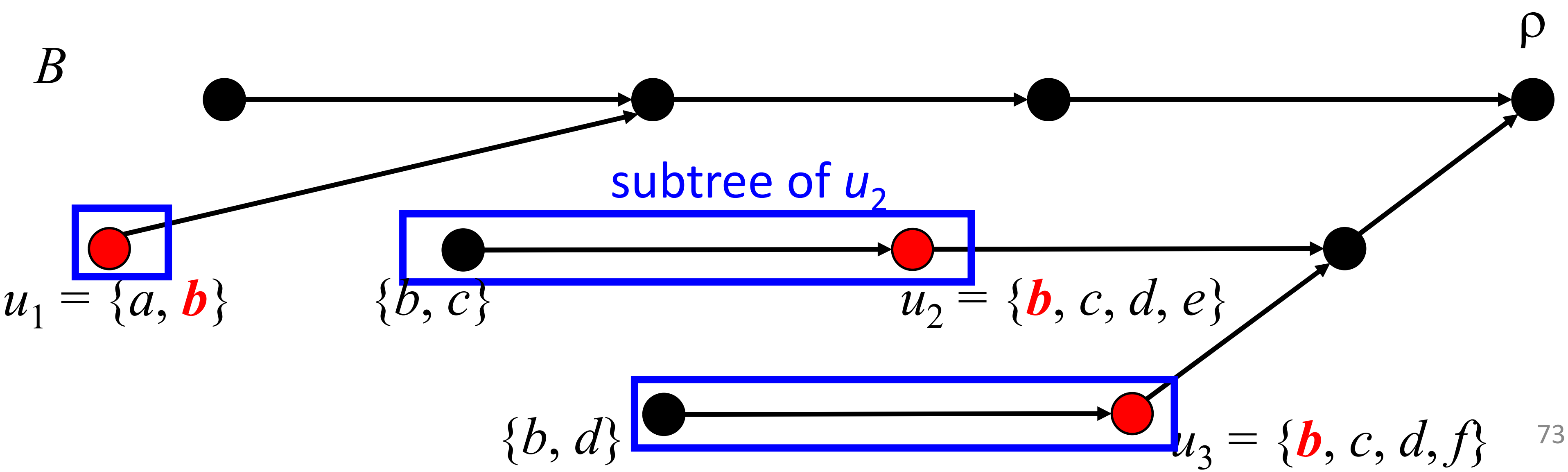
Lemma

- **Lemma 3.1.** If a row r is in every vertex of an antichain X, then $|U_B(r)| \geq |X|$ for any branching B



Proof

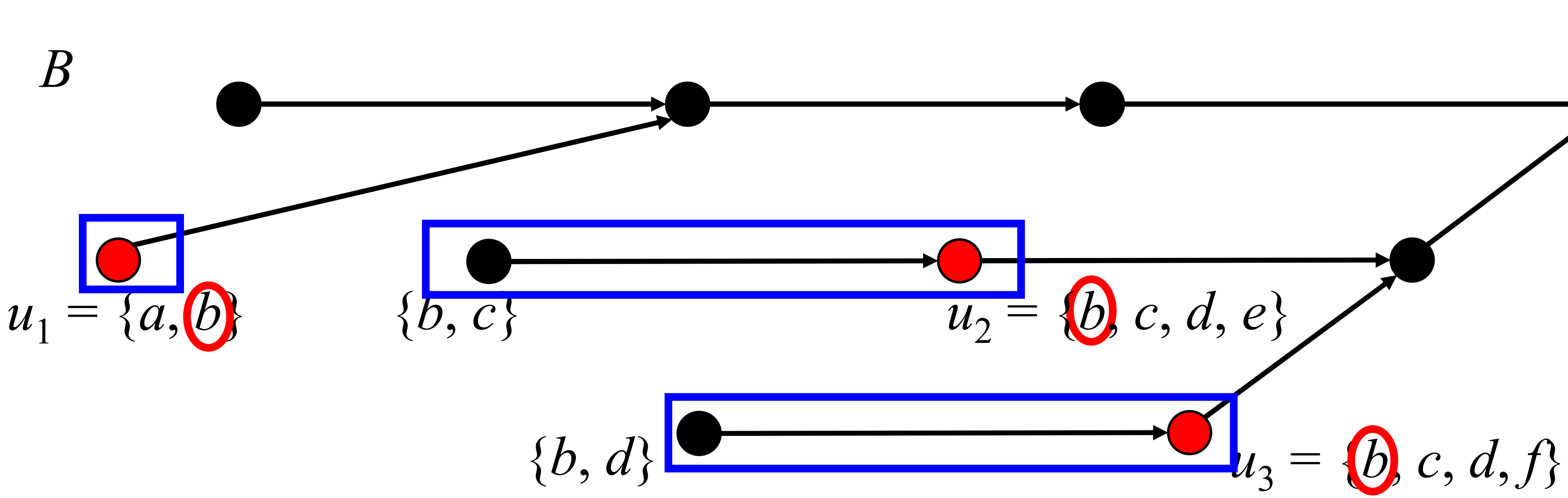
- Let $X = \{u_1, u_2, \ldots, u_{|X|}\}$
- Consider a branching B
- Since X is an antichain, any two vertices u_i, u_j have no ancestor-descendant relation in B.
- Thus, the subtrees of $u_1, u_2, \ldots, u_k$ are disjoint



Proof

- Recall that each u_i corresponds to a target pair (r, u_i)
- Thus, the row r is B-uncovered in some descendant w_i of u_i for each u_i

$\rightarrow U_B(r)$ contains at least $|X|$ uncovered pairs $\square$



Corollary

- **Corollary 3.2.** [12] If r is non-chain-like, then any branching B has $|U_B(r)| \geq 2$.
- **Proof**. By definition, $V_M(r)$ has a non-nested pair (u, v)
- Note that $\{u, v\}$ is an antichain with $r \in u$ and $r \in v$
- By Lemma 3.1, the corollary holds

Lemma

- **Lemma 3.3.** M has at most $\varepsilon(M)$ non-chain-like rows
- **Proof**. Consider an optimal branching B^*,

- Each non-chain-like row r contributes ≥ 2 uncovered pairs

- Thus, each r contributes $|U_{B*}(r)| - 1 \geq 1$ to $\varepsilon(M)$

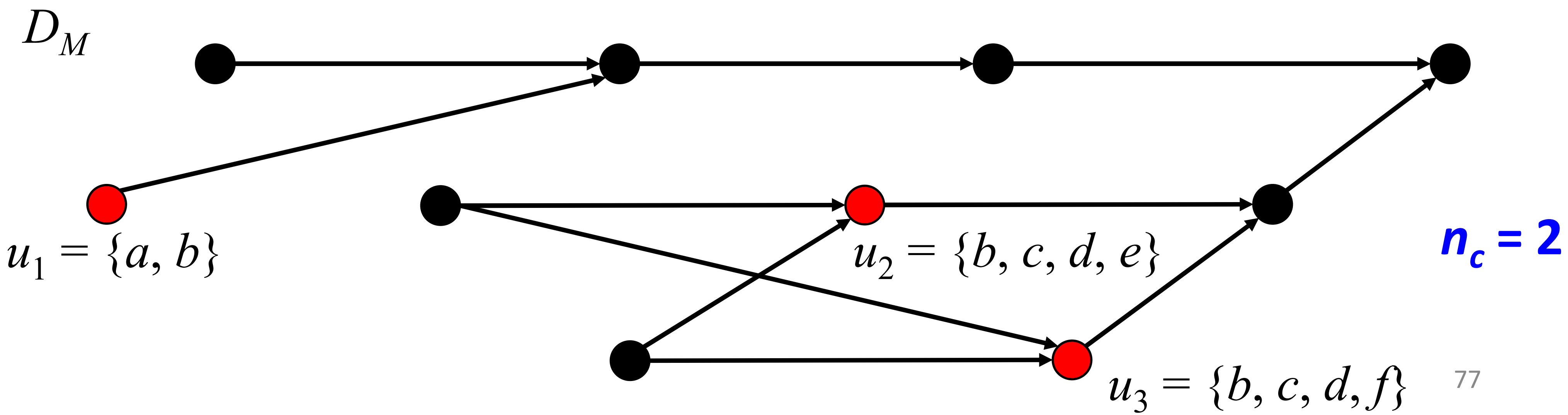
- Therefore, $\varepsilon(M) \geq$ (the number of non-chain-like rows) $\square$

Lemma

- **Lemma 3.5.** The size of any antichain of D_M is at most $2\varepsilon(M)$
- **Proof**. Let

  X: an antichain of D_M

  n_r: the number of vertices containing r in X for each non-chain-like row r of M



Proof

- By Lemma 3.4,
  every vertex in X contains at least one non-chain-like row
- Thus, $\Sigma_{\text{non-chain-like rows } r}\, n_r \geq |X|$

- Let r be a non-chain-like row
- Recall that r is in n_r vertices of X
- By Lemma 3.1, any branching B satisfies $|U_B(r)| \geq n_r$

  Lemma 3.1: r is in every vertex of an antichain $X \rightarrow |U_B(r)| \geq |X|\ \forall\ B$

- Thus, r contributes at least $n_r - 1$ to $\varepsilon(M)$

Proof

- Therefore, we have

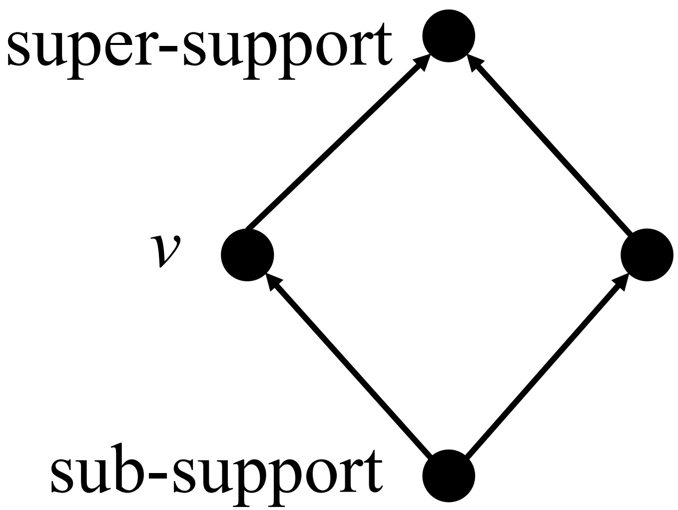
$$\varepsilon(M) \geq \Sigma_{\text{non-chain-like } r} (n_r - 1)$$
$$= \Sigma_{\text{non-chain-like } r}\, n_r - \Sigma_{\text{non-chain-like } r}\, 1$$
$$\geq |X| - \Sigma_{\text{non-chain-like } r}\, 1$$
$$= |X| - (\text{\# non-chain-like rows of } M)$$

- By Lemma 3.3, there are at most $\varepsilon(M)$ non-chain-like rows
- Thus, (# non-chain-like rows of M) $\leq \varepsilon(M)$

$$\rightarrow \varepsilon(M) \geq |X| - \varepsilon(M)$$
$$\rightarrow 2\varepsilon(M) \geq |X| \qquad \square$$

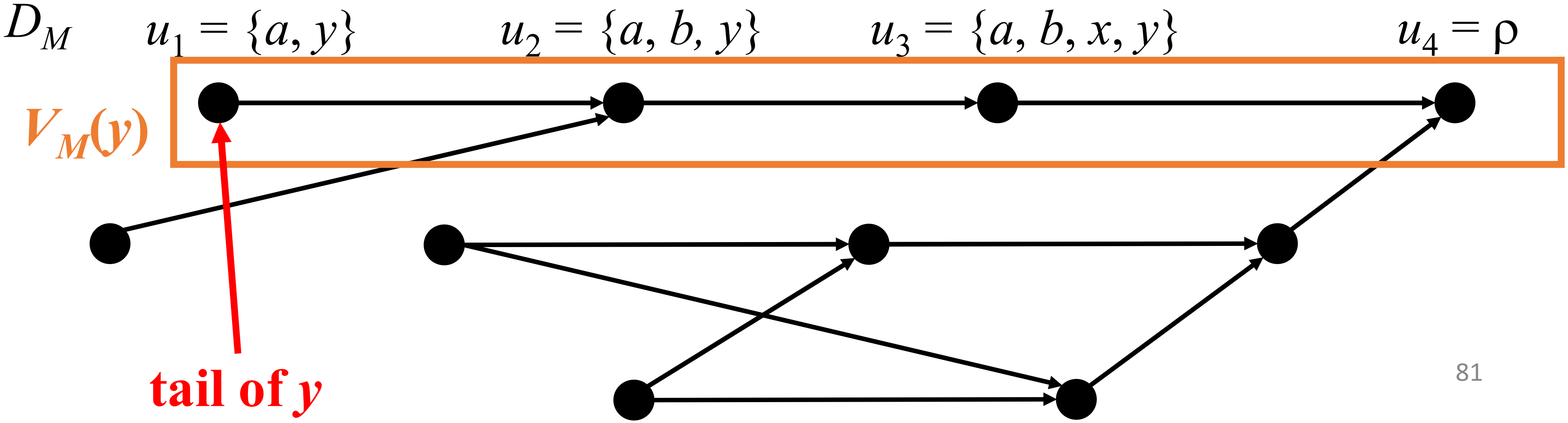
Definition

- Let v be a vertex of D_M
- A *super-support* of v is a vertex u with $v \subset u$
- A *sub-support* of v is a vertex u with $u \subset v$



Definition

- The *tail* of a chain-like row r is the smallest vertex containing r
- Denote by t_r the tail of r
- Note that $V_M(r)$ can be equivalently re-defined as the super-supports of t_r



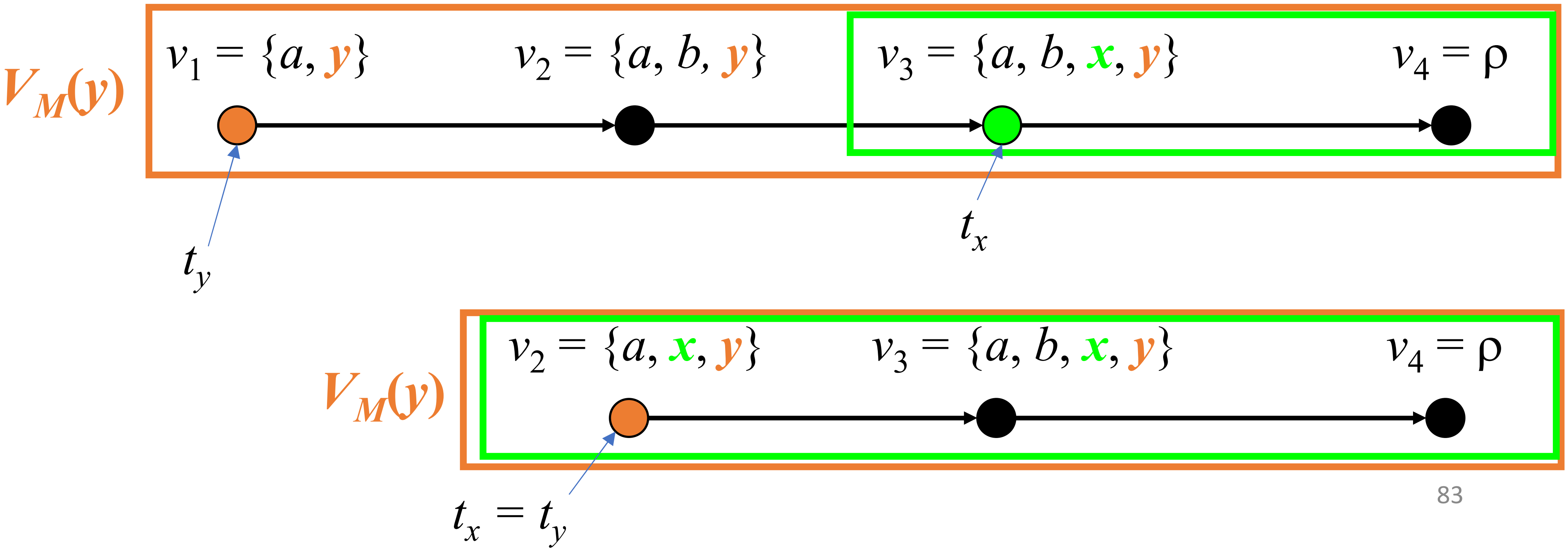
Theorem 3.6

- Let $R(M)$ be the set of rows of M
- **Theorem 3.6.** $|R(M)| \leq 3\varepsilon(M)$
- **Proof.**
- By Lemma 3.3, M has at most $\varepsilon(M)$ non-chain-like rows

- It suffices to show that
  M has at most $2\varepsilon(M)$ chain-like rows

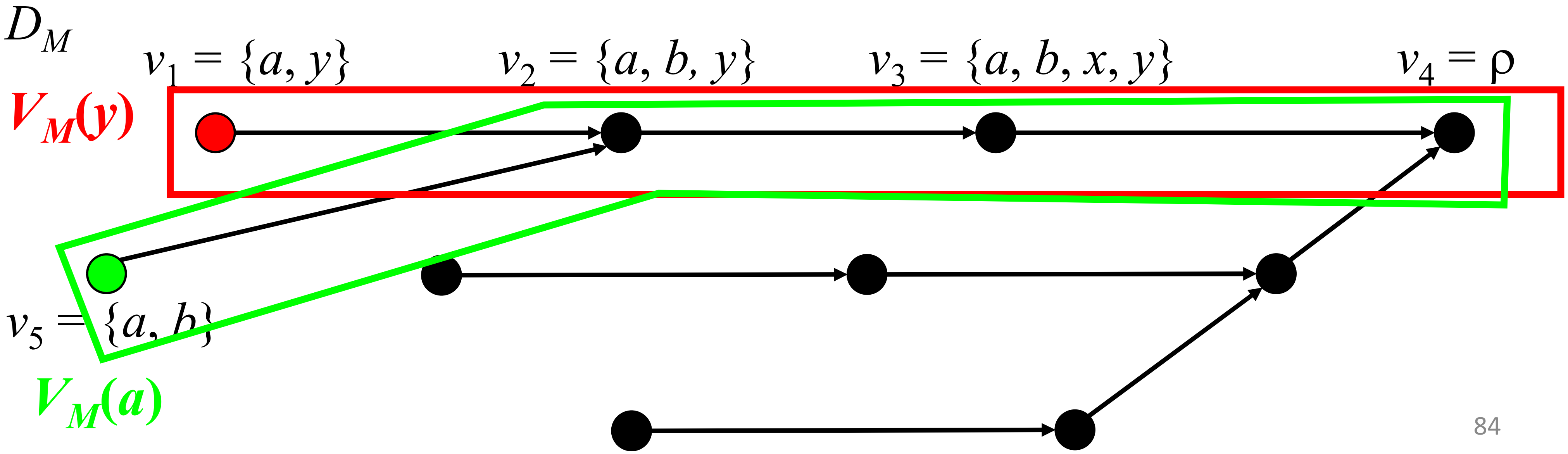
Proof

- Observe that for two chain-like rows r_1, r_2, if $t_{r_1} \subseteq t_{r_2}$, then r_1 is doubly-chain-like



Proof

- Since M has no doubly-chain-like row, the tail of any two chain-like rows are distinct
- In addition, the set $X = \{t_r \mid r \text{ is chain-like}\}$ is an antichain



Proof

- Therefore, X is an antichain with
  $|X|$ = (the number of chain-like rows)

- By Lemma 3.5, $|X| \leq 2\varepsilon(M)$

- Thus, the theorem holds $\square$

Proof

- Let $C(M)$ be the set of columns of M
- **Theorem 3.7.** $|C(M)| \leq 4\varepsilon(M) - 1$

- **Proof.** Recall that $\beta(M) = |R(M)| + \varepsilon(M)$

  $\beta(M)$: the cost of an optimal branching

- By Theorem 3.6, $\beta(M) = |R(M)| + \varepsilon(M) \leq 3\varepsilon(M) + \varepsilon(M)$

- Thus, it suffices to show that $|C(M)| \leq \beta(M) - 1$

Proof

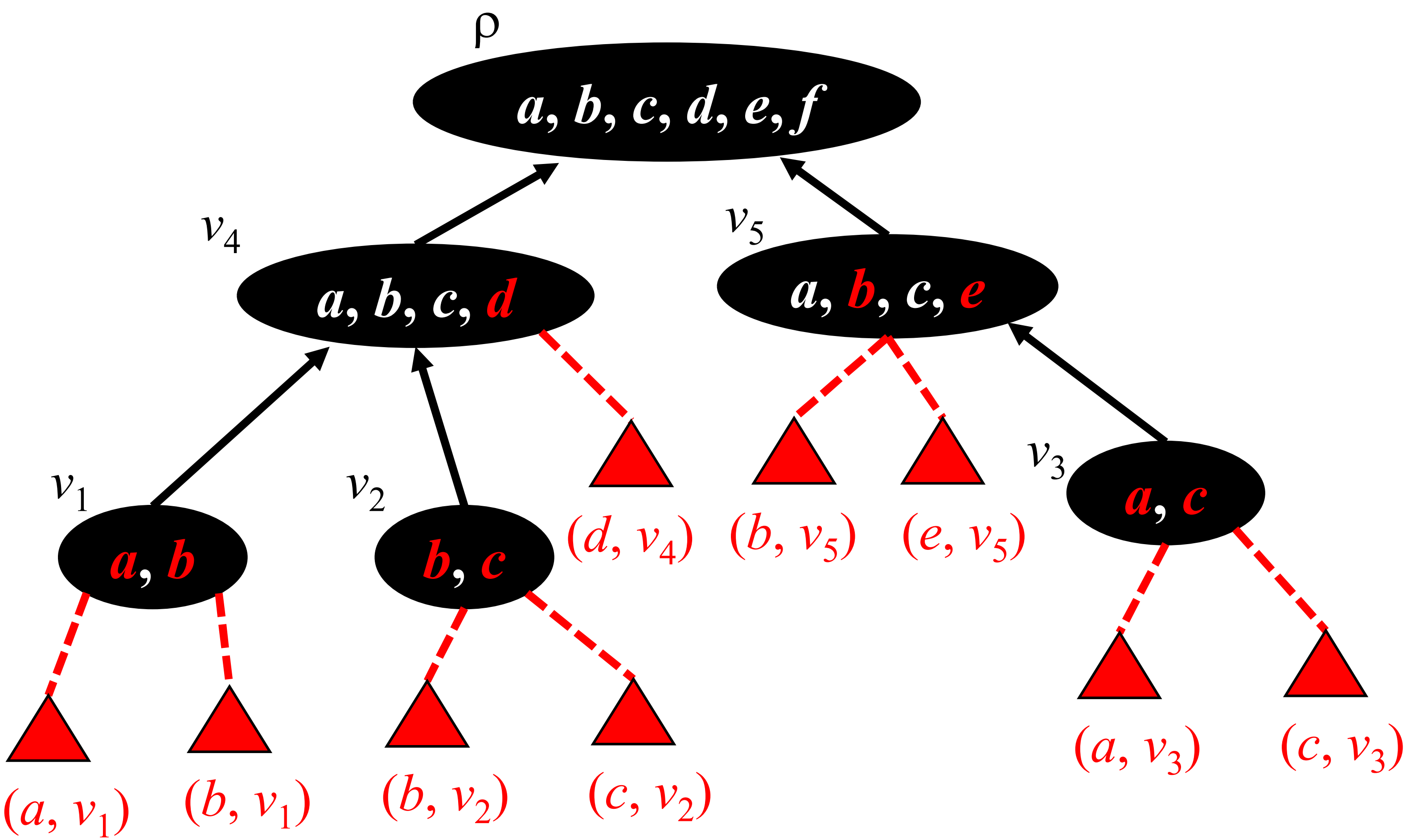
- Let B^* be an optimal branching in which each vertex $v \neq \rho$ has a parent

- Let $F_{B*} = (V(D_M), B^*)$ be a tree rooted at ρ

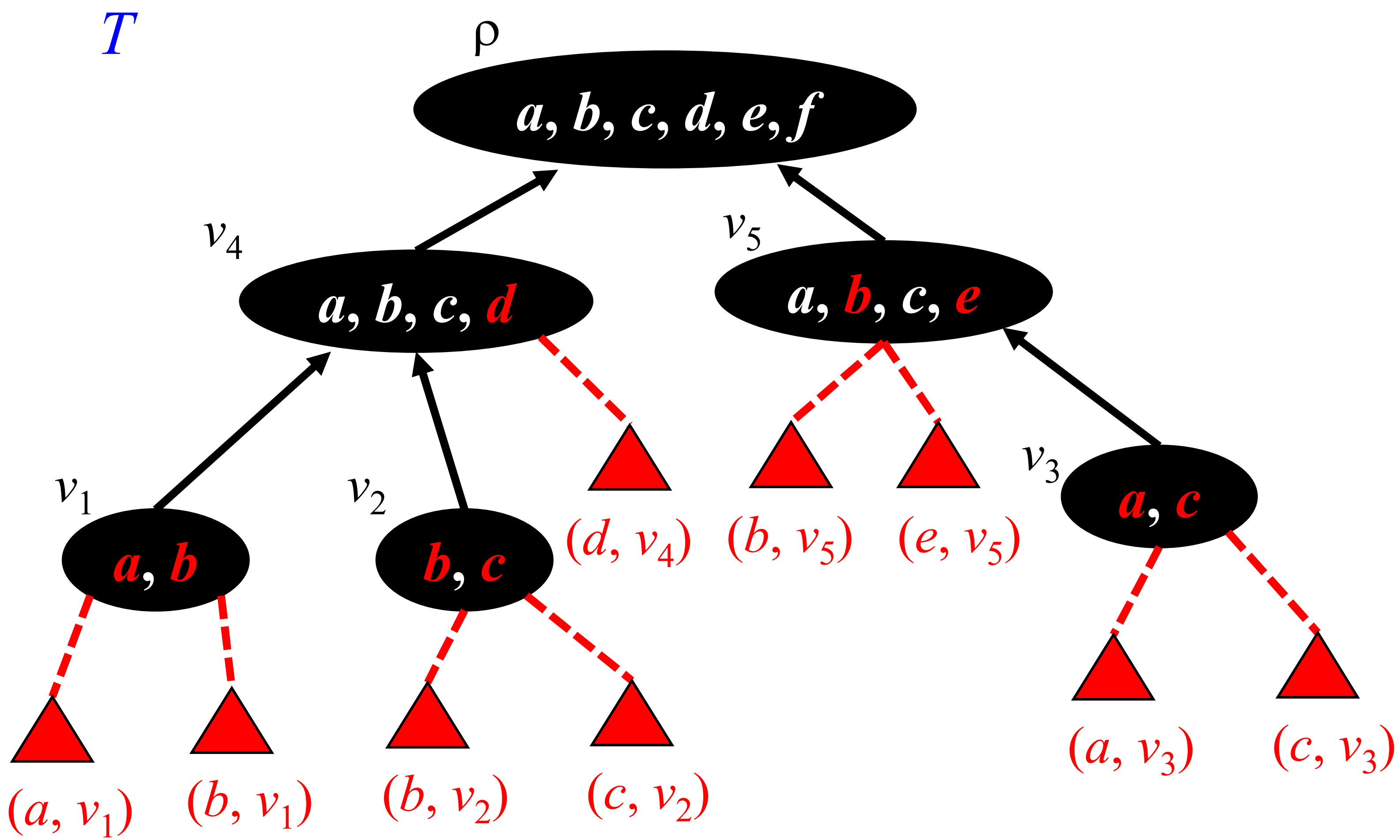
- We construct a tree T from F_{B*} as follows:
  - For each B^*-uncovered pair (r, v), add a leaf child (r, v) of v

- We present an example in the next page

The rooted tree F_{B*}

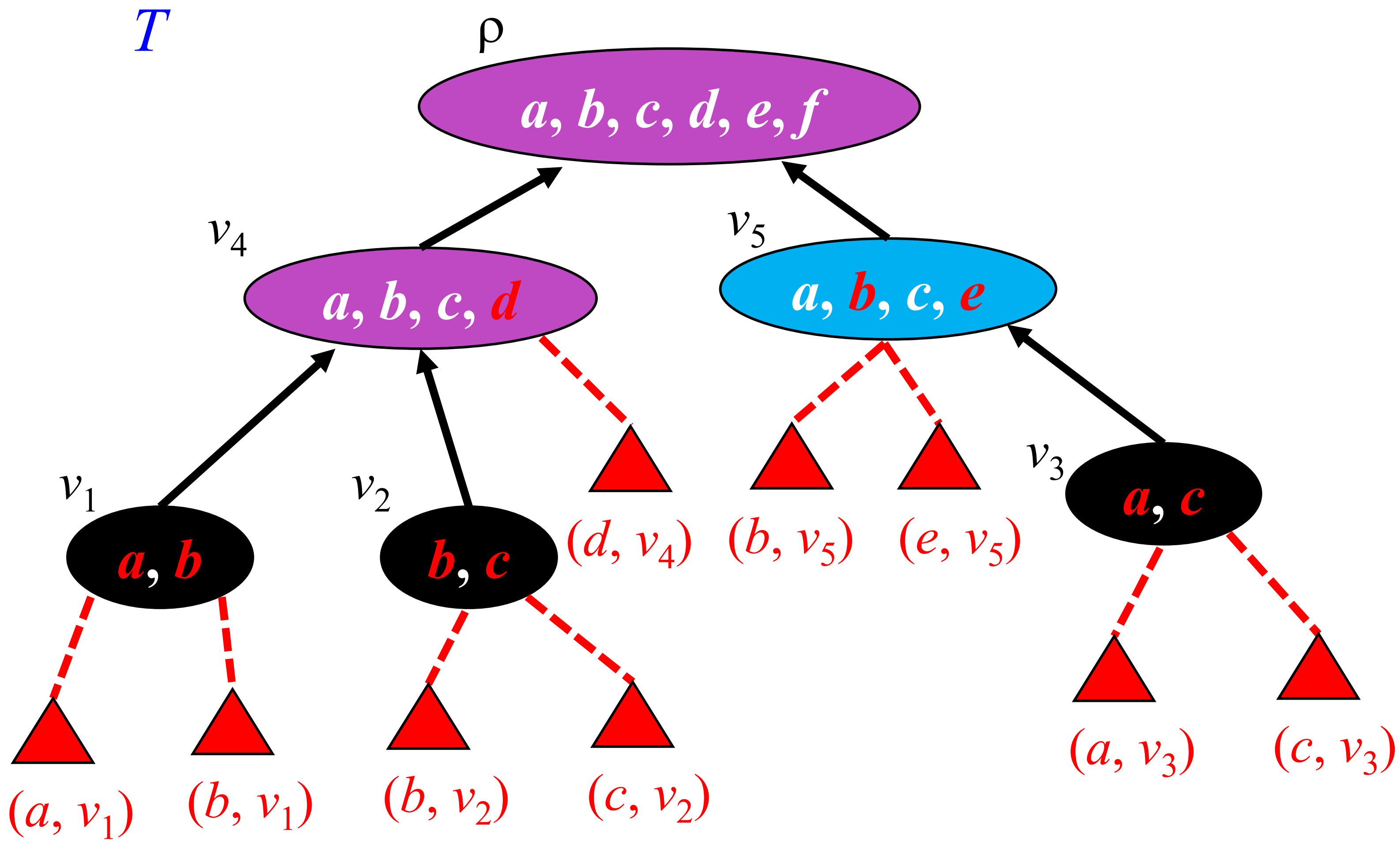


For each uncovered-pair (r, v), add a leaf child (r, v) of v



Note that T is a rooted tree with:

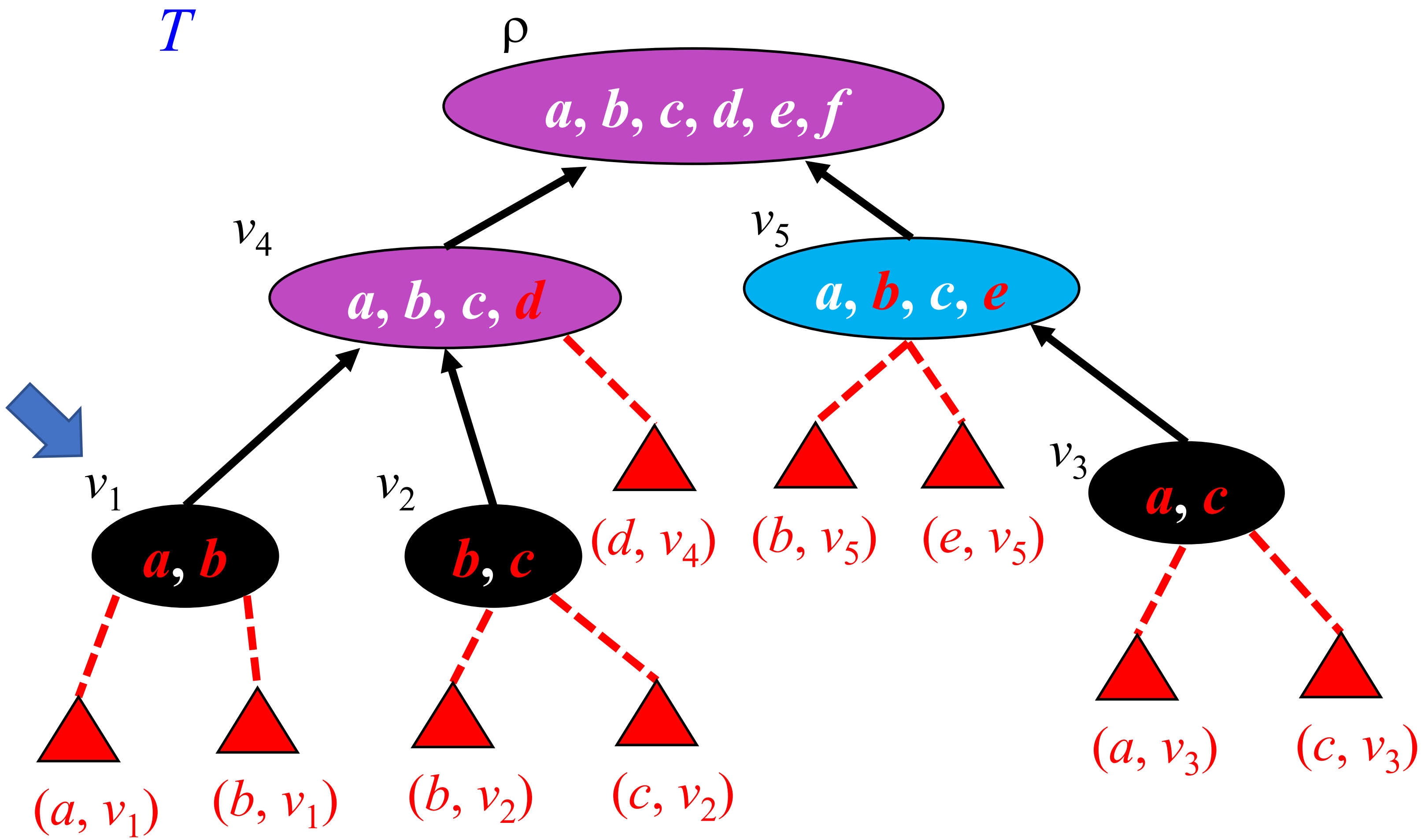
- set of internals $V(D_M)$
- set of leaves $U(B^*)$



We claim that each internal of T has at least two children

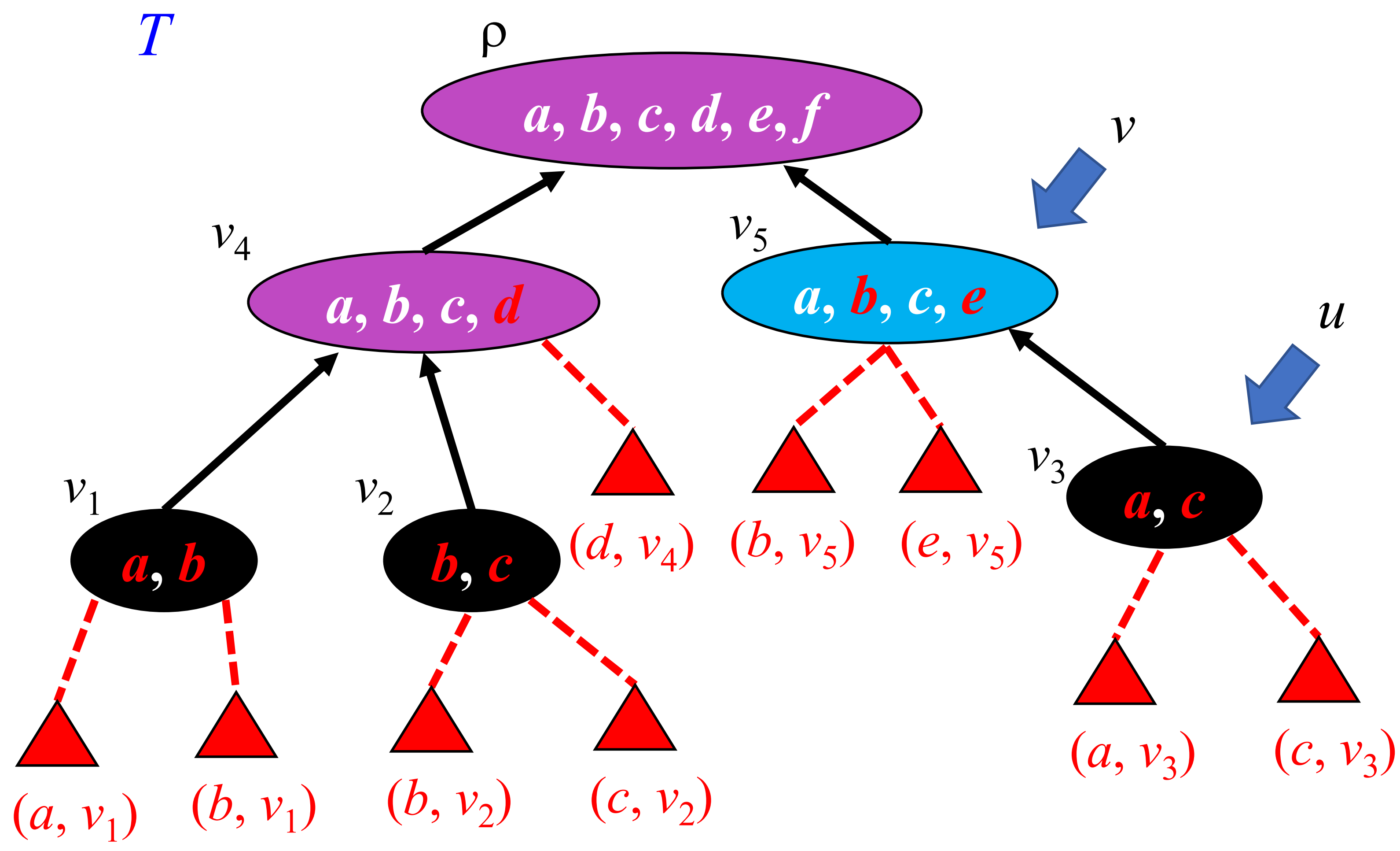
Let $v \in V(D_M)$ be an internal of T

- Case 1: v has in-degree = 0 in B^*
- Case 2: v has in-degree = 1 in B^*
- Case 3: v has in-degree > 1 in B^*



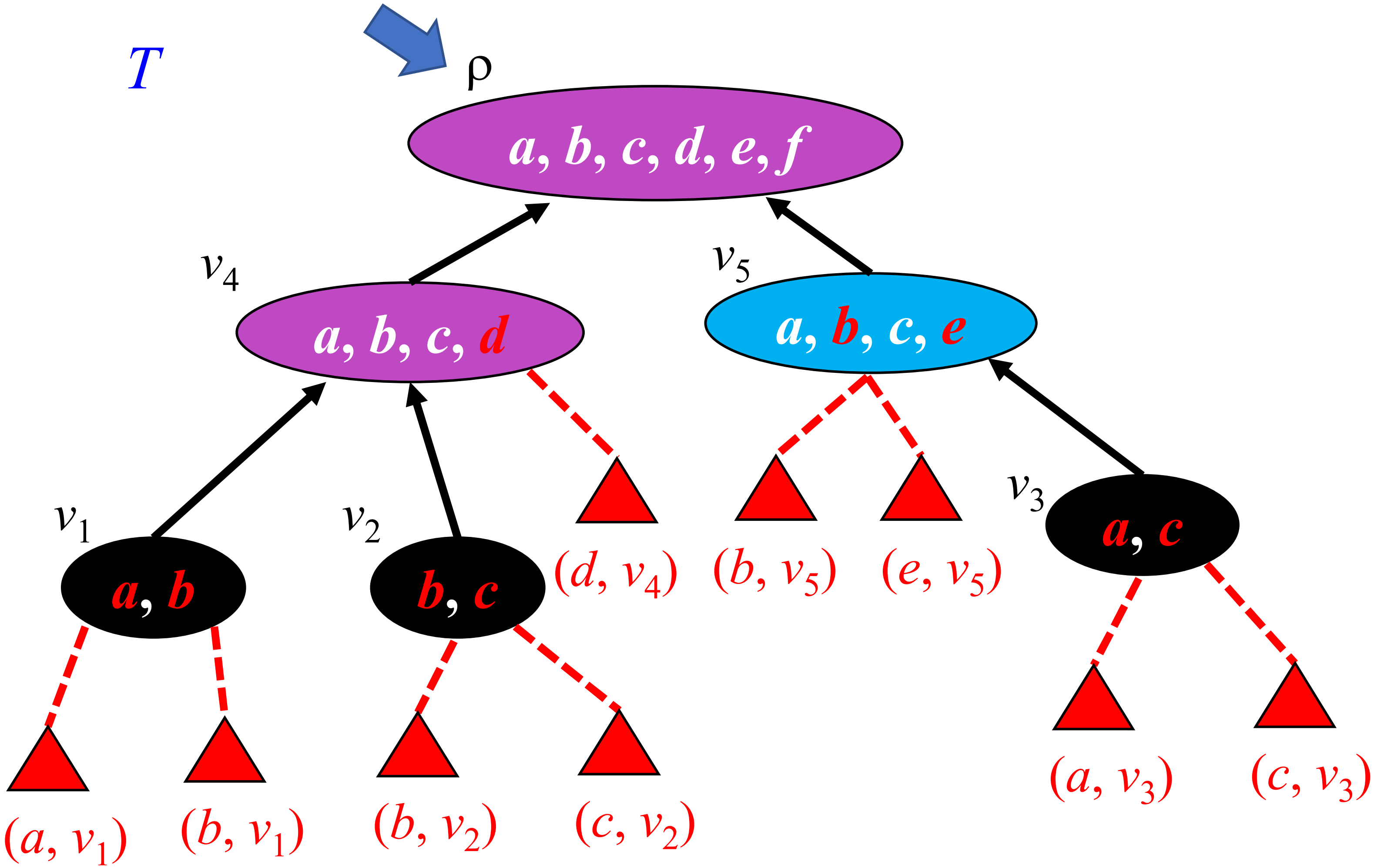
● Case 1: v has in-degree = 0 in B^*

- By Lemma 3.4, $|v| \geq 2$
- Thus, v has ≥ 2 children representing the uncovered elements in v



Case 2: v has in-degree = 1 in B^*

- Let u be the in-neighbor of v in B^*
- Since $u \neq v$, there is at least one row r uncovered in v
- Thus, v has two children: u and the uncovered pair (r, v)



Case 3: v has in-degree > 1 in B^*

Since the in-neighbors of v in B^* are children of v in T, v has at least two children

Proof

- Since each internal of T has two or more children,
  (# internals of T) $\leq$ (# leaves of T) $- 1$

- That is, $|V(D_M)| \leq |U(B^*)| - 1 = \beta(M) - 1$

- This can be rephrased as $|C(M)| \leq 4\varepsilon(M) - 1$

- This completes the proof $\square$

Summary

- By Theorems 3.6 and 3.7, M has at most:
  - $3\varepsilon(M)$ rows
  - $4\varepsilon(M) - 1$ columns

- Thus, the kernel size is upper bounded by $12\varepsilon(M)^2 - 3\varepsilon(M)$

Remark

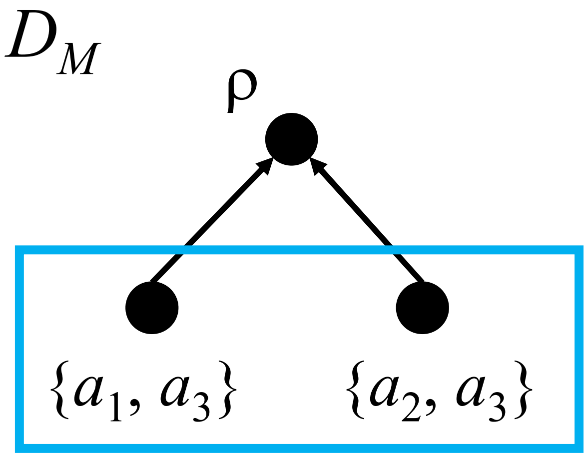
- We remark that the bound on $|R(M)|$ is tight

- Given a positive integer k, a reduced matrix M_k with:
  - $\varepsilon(M) = k$
  - $3\varepsilon(M)$ rows (tight)
  - $2\varepsilon(M) + 1$ columns (not tight)

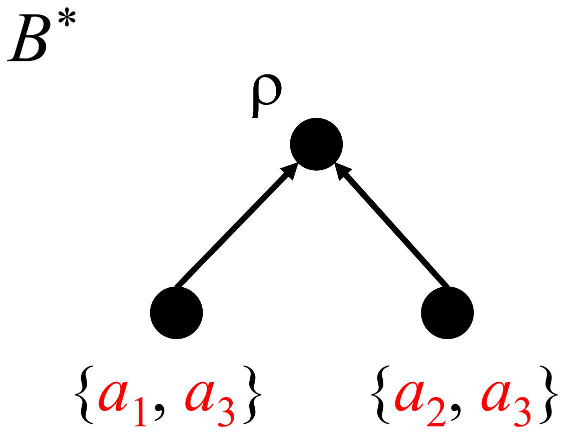
  can be constructed

Construction

- Base case: $k = 1$

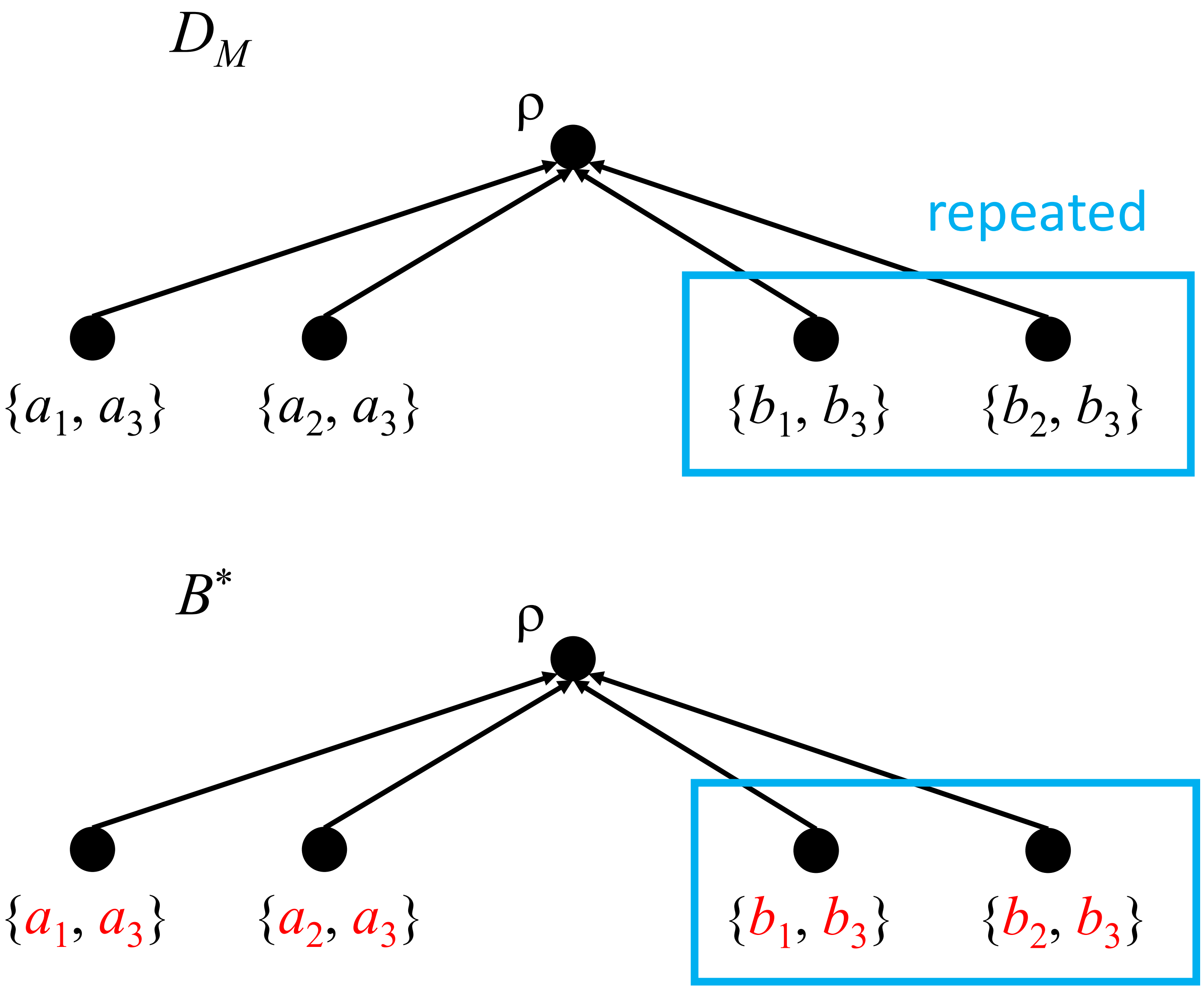


construction: repeat this part



$\beta(M) = 4$
$\varepsilon(M) = 1$
$|R(M)| = 3$
$|C(M)| = 3$

Construction



$\varepsilon(M)$: +1
$|R(M)|$: +3
$|C(M)|$: +2

Outline

- A kernelization algorithm for MSRP
  - Definition
  - Kernel size
  - Safeness
  - The algorithm

Known results

- The *safeness* of Rules 1 and 3 has been shown in [8, 12]

- These results are reviewed as follows

- **Rule 1.** If M contains a pair of duplicate columns c_i, c_j, remove one of them.

- **Lemma 4.1.** [8] Let M^- be a matrix obtained from M by removing a duplicate column, then $\varepsilon(M) = \varepsilon(M^-)$.

Known results

Rule 3. If

(1) Rule 1 is not applicable to M, and

(2) D_M contains a sibling-compatible vertex v

then remove the column c with $supp_M(c) = v$

Lemma 4.2. [12] Let M be a matrix with distinct columns and M^- be a matrix obtained from M by removing a column whose support is a sibling-compatible vertex of D_M, then $\varepsilon(M) = \varepsilon(M^-)$.

- Therefore, it suffices to show the safeness of Rule 2

Notation

- **Rule 2:** If M has a doubly-chain-like row, remove it.

- **Notation**:
- M: a matrix
- M^{-}: obtained by a single application of Rule 2 on M

- x: the doubly-chain-like row removed by the application of Rule 2
- y: a chain-like row such that $V_M(x) \subseteq V_M(y)$
- The rows of M is also labeled by $\{r_1, r_2, \ldots, r_m\}$

Notation

- Label the columns of M and M^{-} with $\{c_1, c_2, \ldots, c_n\}$
- Label the rows of M^{-} with $R(M) - \{x\}$
- Note: each of M and M^{-} may contain duplicate columns

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

M^{-}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

Lemma

- Let s_i denote $supp_M(c_i)$
- Let s'_i denote $supp_{M-}(c_i)$

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

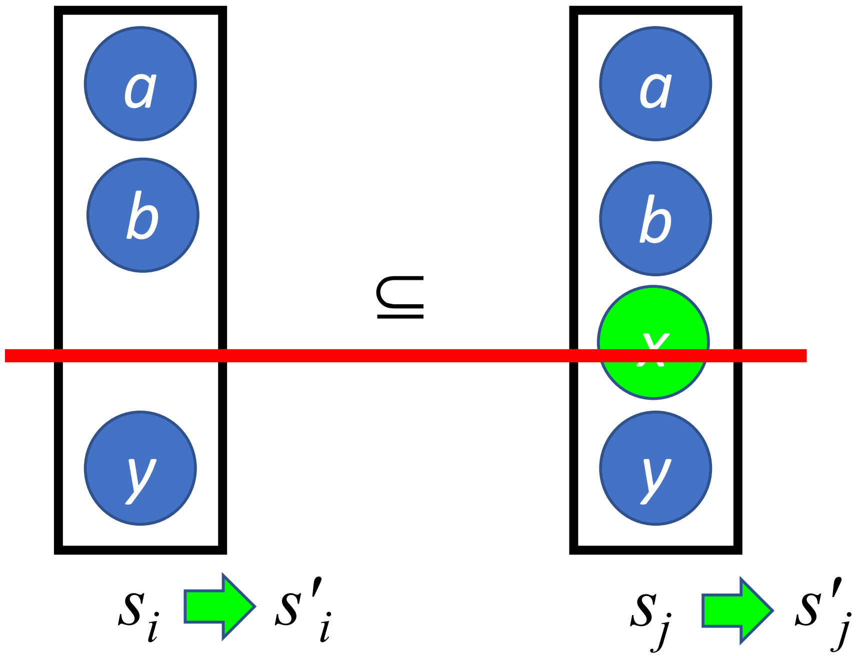
$s_3 = \{r_1, r_2, y, x\}$

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

$s'_3 = \{r_1, r_2, y\}$

Lemma

- The following lemma shows that
  "Rule 2 "preserves" nested relations
- **Lemma 4.3.** For all s_i, s_j, if $s_i \subseteq s_j$, then $s'_i \subseteq s'_j$
- **Proof.** Recall that M^- is obtained by removing x
- If $s_i \subseteq s_j$, then $s_i - \{x\} \subseteq s_j - \{x\}$
- Thus, $s'_i \subseteq s'_j$



Definition

- Let $C_M(r)$ be the set of columns c with $r \in supp_M(c)$

$C_M(r_1) = \{c_2, c_3, \ldots, c_8\}$

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

Recall

- Recall that a row r is chain-like if
  the vertices in $V_M(r)$ are pairwise nested
- Equivalently, a row r is chain-like if
  the columns in $C_M(r)$ are pairwise nested

	M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
	r_1		1	1	1	1	1	1	1
	r_2	1	1	1	1	1	1		1
	r_3					1	1	1	
nested	y	1	1	1	1	1			
	x			1	1	1			

Notation

- W.L.O.G., assume that:
  (1) $C_M(y) = \{c_1, c_2, ..., c_k\}$ where $k = |C_M(y)|$
  (2) $supp_M(c_1) \subseteq supp_M(c_2) \subseteq ... \subseteq supp_M(c_k)$
- By (2), $C_M(x) = \{c_q, c_{q+1}, ..., c_k\}$ for some $q \le k$

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

Corollary

- **Corollary 4.4.** y is a chain-like row of M^{-}. Furthermore, $supp_{M^-}(c_1) \subseteq supp_{M^-}(c_2) \subseteq \ldots \subseteq supp_{M^-}(c_k)$
- **Proof.** Since $supp_M(c_1) \subseteq supp_M(c_2) \subseteq \ldots \subseteq supp_M(c_k)$, by Lemma 4.3, the Corollary is true

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

Known: nested ($\subseteq$)

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

Corollary 4.4: nested ($\subseteq$)

Recall

- The remaining part of the proof uses the original formulation of MSRP
- Recall that a split-row operation splits a row r into several rows whose bitwise OR is r
- The cost is the number of additional rows

r_3			1	1	1	1

cost = 2

E			1			1
B					1	
D				1		1

Definition

- For a matrix P, a *row split* of P is a matrix obtained by performing split-row operations on P

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Definition

- A matrix is *conflict-free* if the supports of any two columns are compatible (nested or disjoint)

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Known result

- MSRP (an equivalent formulation [8, 9, 12]):
- Find a conflict-free row split N of M such that the the number of *additional rows* in N is minimized

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Definition

- For a row split Q of P, a *feasible partition of Q* (with respect to P) is a partition of rows of Q into $m = |R(P)|$ sets $Q|^{r_1}, Q|^{r_2}, \ldots, Q|^{r_m}$ such that the bitwise OR of the rows in $Q|^{r_i}$ is r_i

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

$N|^{r_1}$

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Definition

- As in [8, 12], we make a slight technical abuse by considering any row split Q of P as already equipped with an fixed feasible partition (with respect to P)

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

$N|^{r_1}$

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Notation

- For ease of notation, we assume that the columns of any row split of M (resp. M^{-}) are labeled with $\{c'_1, c'_2, \ldots, c'_n\}$ such that c'_i is the corresponding column of c_i

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Lemma

- **Lemma 4.5.** [12] For any matrix P,
  there exists an optimal conflict-free row split Q s.t.
  for each chain-like row r of P,
  $Q|^r$ contains a single row identical to r

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

Condition: x is chain-like

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Result: identical to x

Lemma

- **Lemma 4.6.** There exists an optimal conflict-free row split N^- of M^- satisfying the following:
  - (P1) $N^-|^y$ contains a single row identical to y
  - (P2) $supp_{N^-}(c'_1) \subseteq supp_{N^-}(c'_2) \subseteq \ldots \subseteq supp_{N^-}(c'_k)$

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

Known: nested ($\subseteq$)

N^-	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

Lemma 4.6: nested ($\subseteq$)

Proof

- By Lemma 4.5, there exists an optimal solution satisfying (P1)

(P1) $N^{-}|y$ contains a single row

- Among all such solutions, let N^{-} be the one having the most duplicate columns

- In the following, we show that N^{-} satisfies (P2)

(P2) $supp_{N-}(c'_1) \subseteq supp_{N-}(c'_2) \subseteq \ldots \subseteq supp_{N-}(c'_k)$

Proof

- Consider two columns c_i, c_{i+1}, where $1 \leq i < k$
- Recall that $supp_{M-}(c_i) \subseteq supp_{M-}(c_{i+1})$
- Consider two cases:
    1. $supp_{M-}(c_i) = supp_{M-}(c_{i+1})$ (duplicate)
    2. $supp_{M-}(c_i) \subset supp_{M-}(c_{i+1})$

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

Case 2 **Case 1**

N^-	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

Proof

- If c_i and c_{i+1} is a pair of duplicate columns,
  c'_i and c'_{i+1} must be a pair of duplicate columns
- (Otherwise, replacing c'_i with c'_{i+1} to reach a contradiction)
- Thus, $supp_{N^-}(c'_i) \subseteq supp_{N^-}(c'_{i+1})$

duplicate

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

duplicate

N^-	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

Proof

- Suppose $supp_{M-}(c_i) \subset supp_{M-}(c_{i+1})$
- Let r be a row in $supp_{M-}(c_{i+1}) - supp_{M-}(c_i)$
- In $N^{-}|^{r}$, there is a row r' in $supp_{N-}(c'_{i+1}) - supp_{N-}(c'_i)$
- Since N^{-} is conflict-free, c'_i and c'_{i+1} are nested, and thus $supp_{N-}(c'_i) \subset supp_{N-}(c'_{i+1})$

duplicate

r

M^{-}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

r'

N^{-}	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

Theorem

- **Theorem 4.7.** $\varepsilon(M) = \varepsilon(M^-)$
- **Proof.** We first show that $\varepsilon(M^-) \leq \varepsilon(M)$
- Let N be an optimal conflict-free row split of M
- Let N^- be the matrix obtained from N by removing all rows in $N^-|^x$

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
x			1	1	1			

N^-

$\cancel{N}$	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
~~x'~~			~~1~~	~~1~~	~~1~~			
~~x''~~				~~1~~	~~1~~			

Proof

- To prove $\varepsilon(M^-) \leq \varepsilon(M)$, it suffices to show:
  N^- is a conflict-free row split of M^- with
  at most $\varepsilon(M)$ additional rows
- Clearly, N^- is a row split of M^- with at most $\varepsilon(M)$ additional rows

M^-

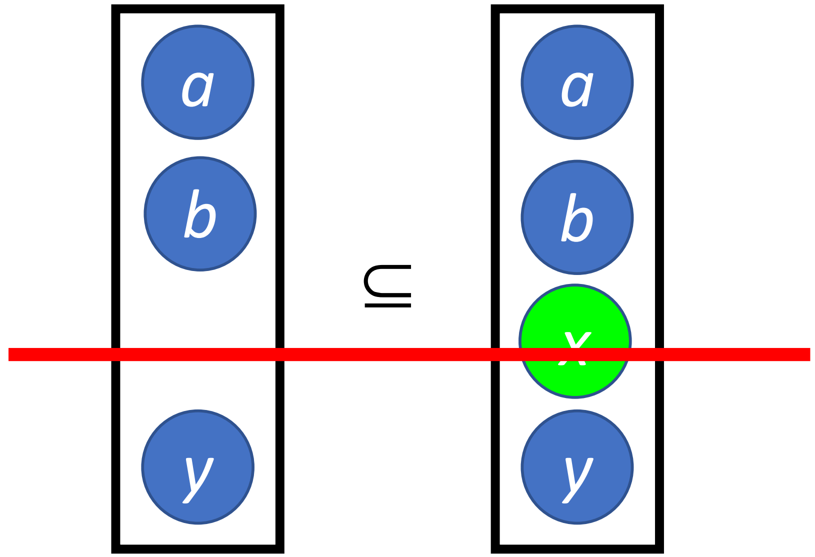
~~M~~	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			
~~x~~			~~1~~	~~1~~	~~1~~			

N^-

~~N~~	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
~~x'~~			~~1~~	~~1~~	~~1~~			
~~x''~~				~~1~~	~~1~~			

Proof

- To show N^{-} is conflict-free, we claim that removing any row r from a conflict-free matrix (e.g. N) does not induce any pair of conflicting columns
- Two disjoint columns of N $\rightarrow$ still disjoint after removing r
- Two nested columns of N
  $\rightarrow$ nested or disjoint after removing r
- Thus, N^{-} is conflict-free and $\varepsilon(M^{-}) \leq \varepsilon(M)$



Proof

- We proceed to show that $\varepsilon(M) \leq \varepsilon(M^-)$
- Let N^- be an optimal conflict-free row split of M^-
- By Lemma 4.6, we may assume that N^- satisfies:
  - (P1) $N^-|^y$ contains a single row identical to y
  - (P2) $supp_{N^-}(c'_1) \subseteq supp_{N^-}(c'_2) \subseteq \ldots \subseteq supp_{N^-}(c'_k)$

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

N^-	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

Proof

- We obtain from N^- a matrix N by appending a row, labeled by x', such that $N_{x',c'_i} = M_{x',c_i}$ for all columns $c_i \in C(M)$

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

N $\cancel{N^-}$	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			**1**	**1**	**1**			

Proof

- To prove $\varepsilon(M^-) \le \varepsilon(M)$, it suffices to show that
  (1) N^- is a row split of M^- with exacyly $\varepsilon(M)$ additional rows (clearly)
  (2) N^- is conflict-free

M^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
r_1		1	1	1	1	1	1	1
r_2	1	1	1	1	1	1		1
r_3					1	1	1	
y	1	1	1	1	1			

N

$\cancel{N^-}$	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Proof

- Let y' be the only row in $N^{-}|^{y}$
- Note that $C_N(y') = \{c'_1, c'_2, \ldots, c'_k\}$
- Consider a pair of columns c'_i, c'_j, where $i < j$
- Let $s^{-}_i = supp_{N-}(c'_i)$, and $s_i = supp_N(c'_i)$

c'_k

N^{-}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Proof

- Three cases are considered:

  Case 1: $i, j > k$ → **support unchanged** → **compatible**

  Case 2: $i, j \le k$

  Case 3: $i \le k, j > k$

c'_k

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Case 2: $i, j \leq k$

- Recall that (P2) $s^-_1 \subseteq s^-_2 \subseteq \ldots \subseteq s^-_k$
- Since $C_N(x') = \{c'_q, c'_{q+1}, \ldots, c'_k\}$, after appending x', we have
  $s_1 \subseteq s_2 \subseteq \ldots \subseteq s_k$

N^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

(P2): nested ($\subseteq$)

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Case 3: $i \le k, j > k$

- In Case 3, $y \in s^-_i$ but $y \notin s^-_j$
- Since N^- is conflict-free, either $s^-_i \supset s^-_j$ or s^-_i, s^-_j are disjoint
- In both cases, s_i and s_j are compatible

N^-	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			

no y'

N	c'_1	c'_2	c'_3	c'_4	c'_5	c'_6	c'_7	c'_8
a_1		1	1	1	1			
a_2					1	1	1	
a_3					1	1		1
b_1	1	1	1	1	1			
b_2					1	1		1
c'					1	1	1	
y'	1	1	1	1	1			
x'			1	1	1			

Proof

- Therefore, N is indeed conflict-free
- As a result, $\varepsilon(M^{-}) \leq \varepsilon(M)$
- This completes the proof.

Outline

- Introduction
- Preliminaries
- A kernelization algorithm for MSRP
- An approximation algorithm for MSRP
- **Approximation algorithms for MDCRSP**
- Conclusion and future work

Recall

- Recall that a conflict-free row split is a feasible solution

- That is, a matrix which
  1. can be obtained by split-row operations, and
  2. corresponds to a perfect phylogeny

Recall

- MDCRSP asks to
  find a conflict-free row split of M
  with the minimum number of distinct rows

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1		1	1	1
r_2					1	
r_3			1	1	1	1
r_4		1				1

M'	c_1	c_2	c_3	c_4	c_5	c_6
$r_1^{(1)}$	1				1	
$r_1^{(2)}$		1				1
$r_1^{(3)}$				1		1
$r_2^{(1)}$					1	
$r_3^{(1)}$			1			1
$r_3^{(2)}$					1	
$r_3^{(3)}$				1		1
$r_4^{(1)}$		1				1

cost = 5 (distinct rows)

Recall

- We will present new approximation algorithms for MDCRSP

Source	Approximation ratio	Time
[8]	2	$O(mn^2)$
[this]	$5/3 \approx 1.67$	$O(mn^2)$
[this]	$4/3 + \delta$ for any $\delta > 0$	$n^{O(1/\delta)} \approx n^{64/\delta}$

Known result

- Let η(M) be the minimum number of distinct rows in a conflict-free row split
- Similar to MSRP, removing duplicate columns does not change η(M) [8]
- Thus, we assume that M has no duplicate columns

M	c_1	c_2	c_3	c_4	c_5	c_6	c_7
r_1	1	1		1	1	1	1
r_2					1		
r_3			1	1	1	1	1
r_4		1				1	1

M	c_1	c_2	c_3	c_4	c_5	c_6
r_1	1	1		1	1	1
r_2					1	
r_3			1	1	1	1
r_4		1				1

Known results

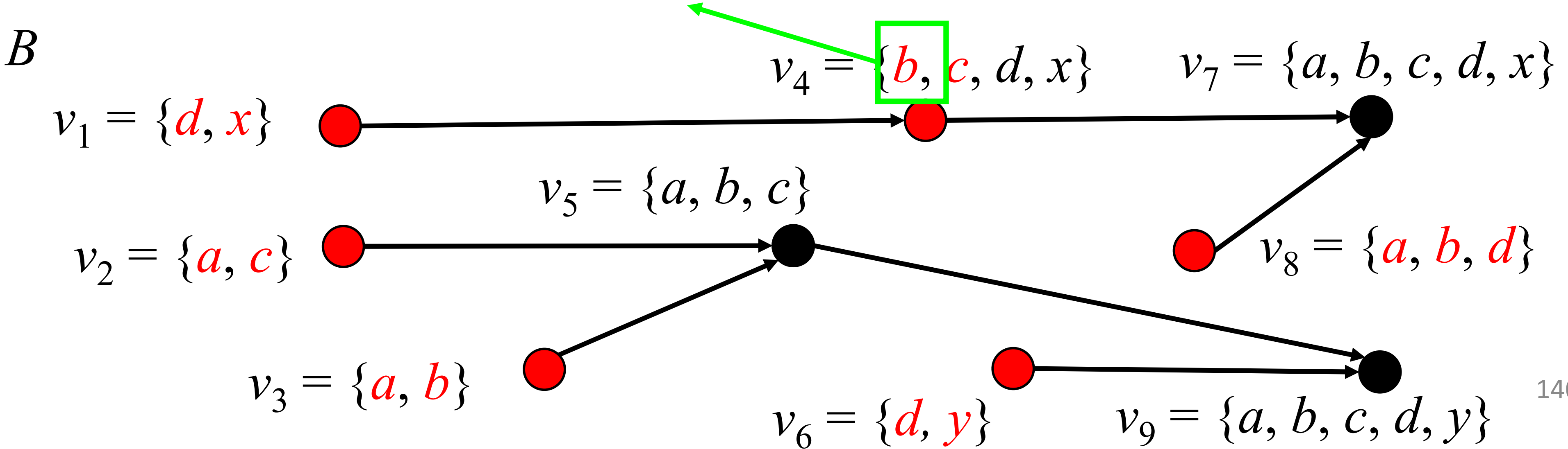
- MDCRSP admits a formulation
  similar to the branching formulation [8]

- Recall: the branching formulation

Definition

- Recall that a *B*-uncovered pair is a target pair (r, v) such that r is not in any child of v
- A vertex v is *B-irreducible* if there is a B-uncovered pair (r, v) for some row r

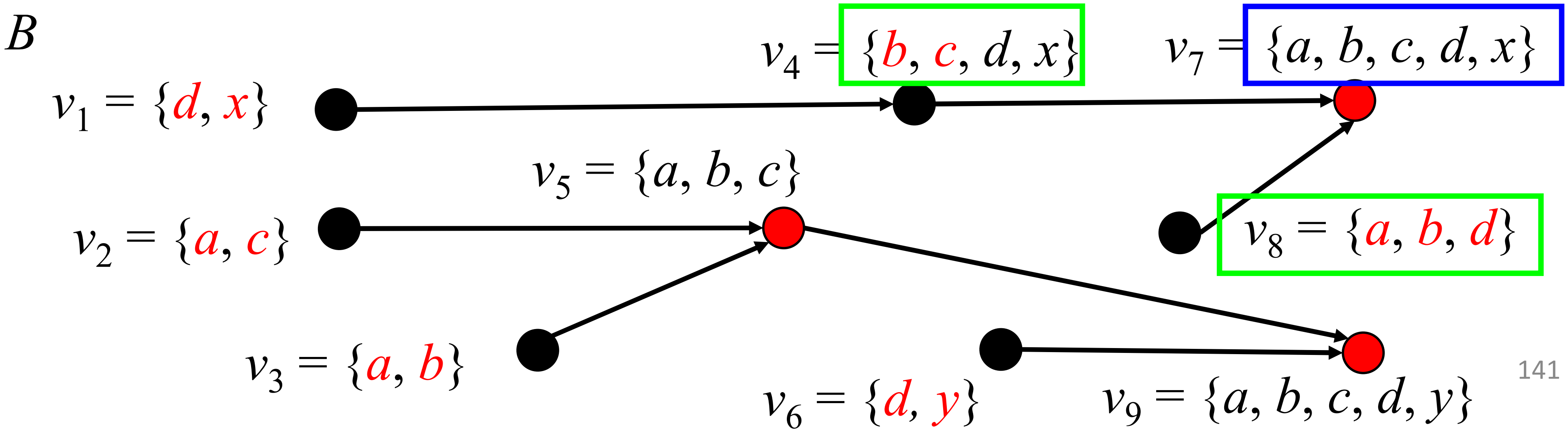
has an uncovered pair (b, v_4) → v_4 is irreducible



Definition

- A vertex v is *B-reducible* if
  it is not B-irreducible

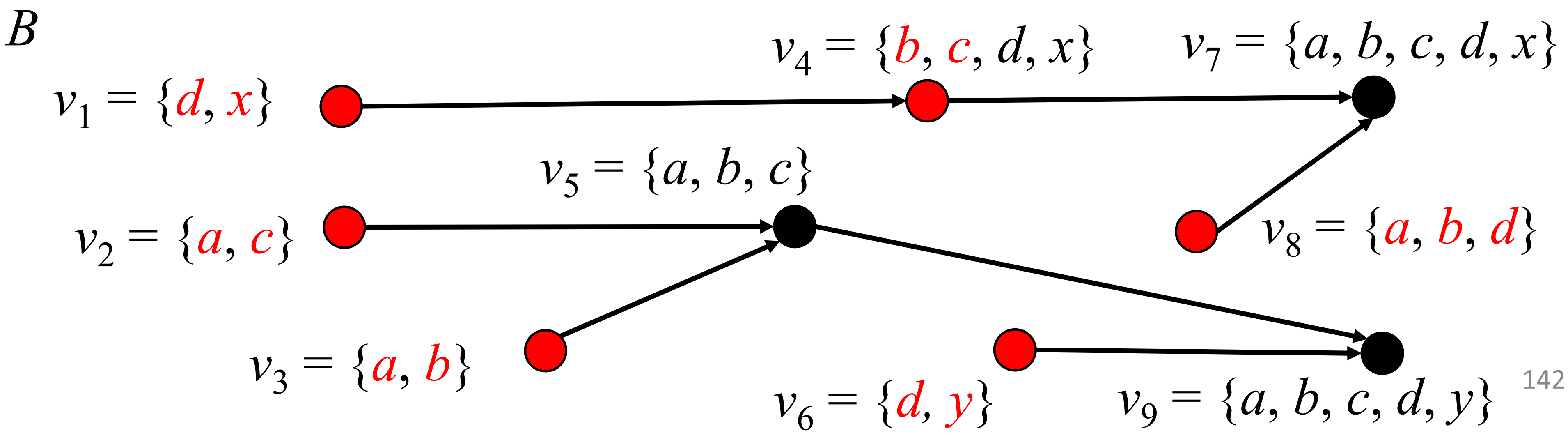
- In other words, a vertex v is B-reducible if
  the union of its B-children is itself



Definition

- Let $I(B)$ be the set of all B-irreducible vertices
- We re-define *cost* of a branching B as $|I(B)|$
- Let $\zeta(M)$ be the minimum cost of a branching

cost = 6



Known result

- MDCRSP is equivalent to
  finding the minimum cost branching [8]

- **Theorem 5.1.** [8] For any matrix M, the following hold:

1. Any branching B of D_M can be transformed to a conflict-free row split with $|I(B)|$ distinct rows.
2. Any conflict-free row split M' of M can be transformed to a branching B such that $|I(B)|$ is at most the number of distinct rows of M'.

- Consequently, $\eta(M) = \zeta(M)$

Remark

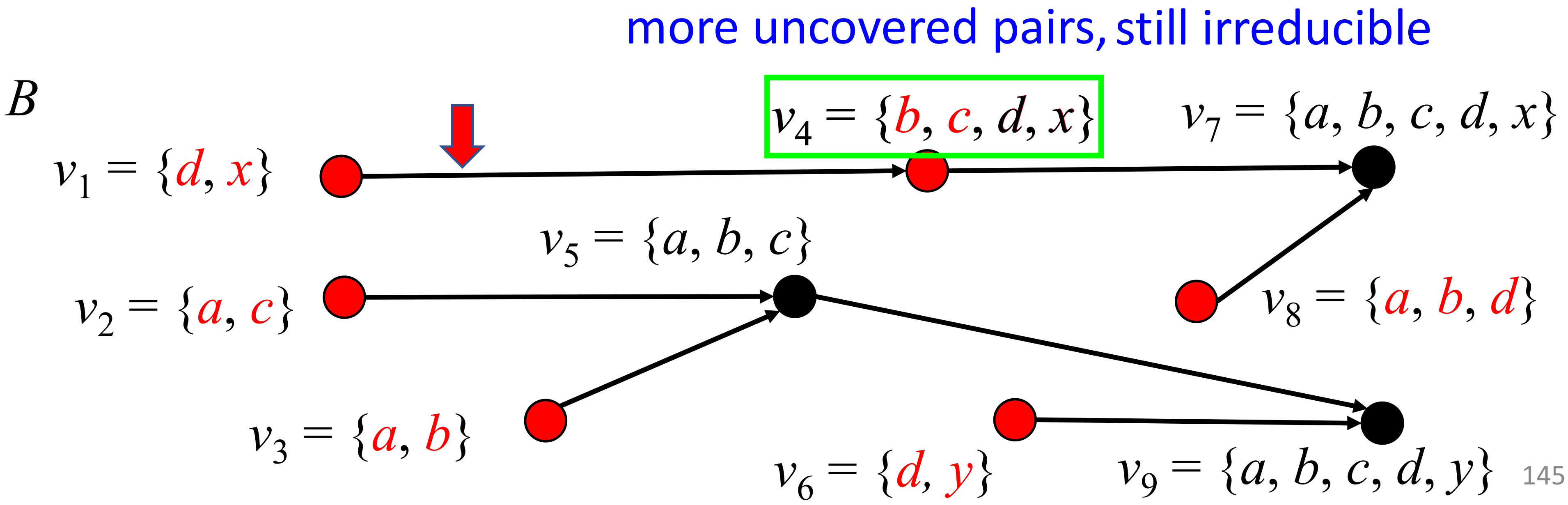
- By Theorem 5.1, the approximation of $\eta(M)$ can be done by approximating $\zeta(M)$

- We begin by simple observations

- Consider a branching B

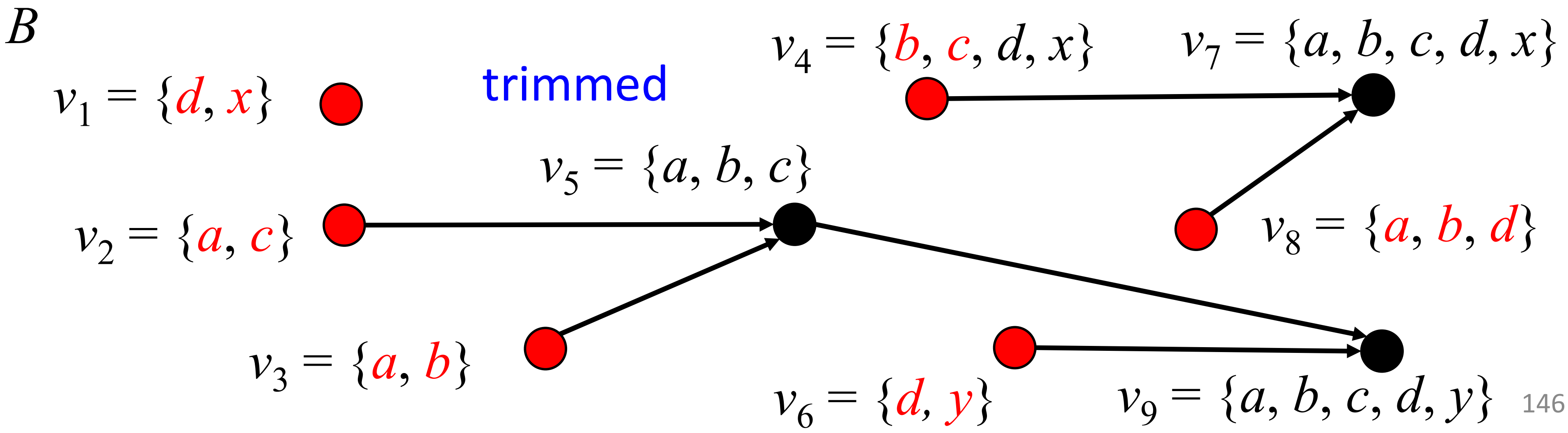
Observation

- Suppose that there is an arc $(v, v') \in B$ s.t. v' is irreducible
- Let $B' = B - \{(v, v')\}$
- The costs of B and B' are the same



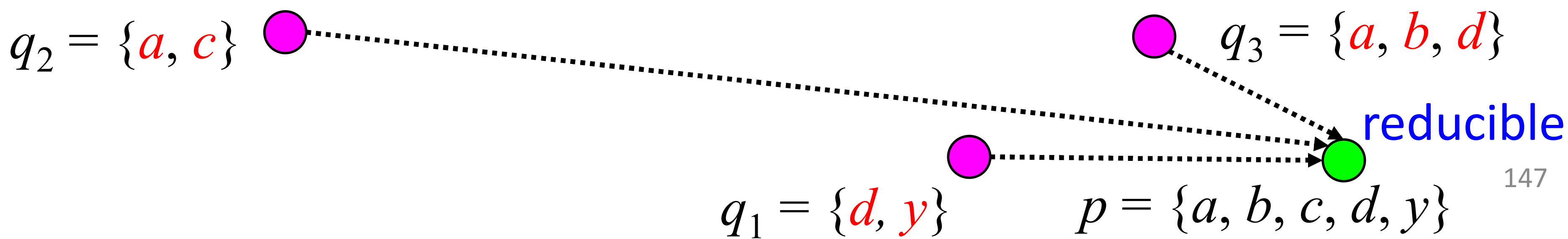
Definition

- A branching is *trimmed* if
  every irreducible vertex has no child
- By the observation, each branching B can be transformed to
  a trimmed branching B' with $I(B) = I(B')$
- In MDCRSP, it suffices to consider trimmed branchings



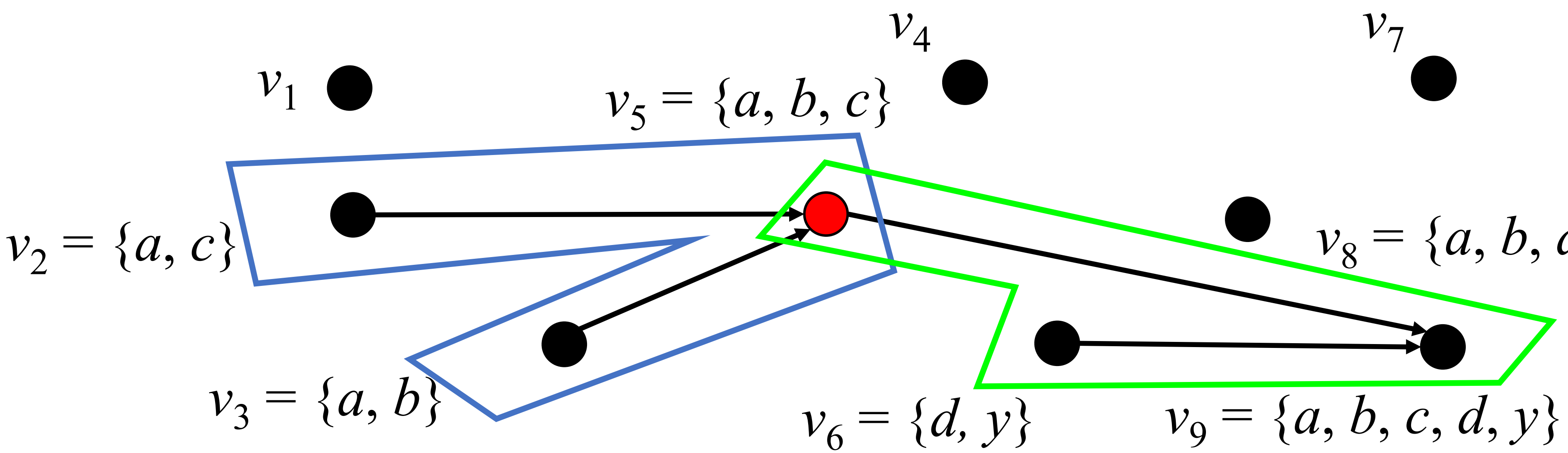
Definition

- A *candidate* is a pair (p, Q), where
  p is a vertex (of D_M) and Q is a subset of vertices, such that
  (1) $B = \{(v, p) \mid v \in Q\}$ is a branching
  (2) p is B-reducible



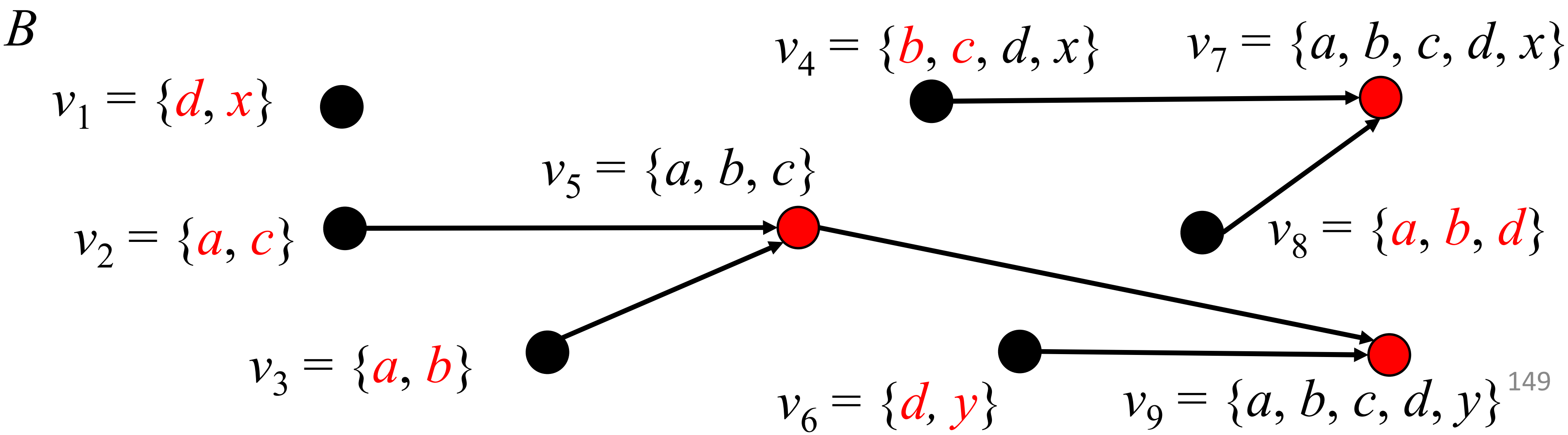
Definition

- Two candidates (p_1, Q_1), (p_2, Q_2) are *disjoint* if
  1. $p_1 \neq p_2$
  2. Q_1 and Q_2 are disjoint
- Note that Q_1 may contain p_2 (resp???)
- Example: $(v_5, \{v_2, v_3\})$ and $(v_9, \{v_5, v_6\})$



Observation

- A trimmed branching B can be transformed to a set of pairwise disjoint candidates:
- Each reducible vertex p corresponds to a candidate (p, Q) where Q is the child set of p
- Since B has $n - |I(B)|$ reducible vertices, the resulting set contains $n - |I(B)|$ candidates



Observation

- Recall that MDCRSP seeks a branching with the minimum number of irreducible vertices

- Thus, the exact solution of MDCRSP can be found by selecting the maximum number of disjoint candidates

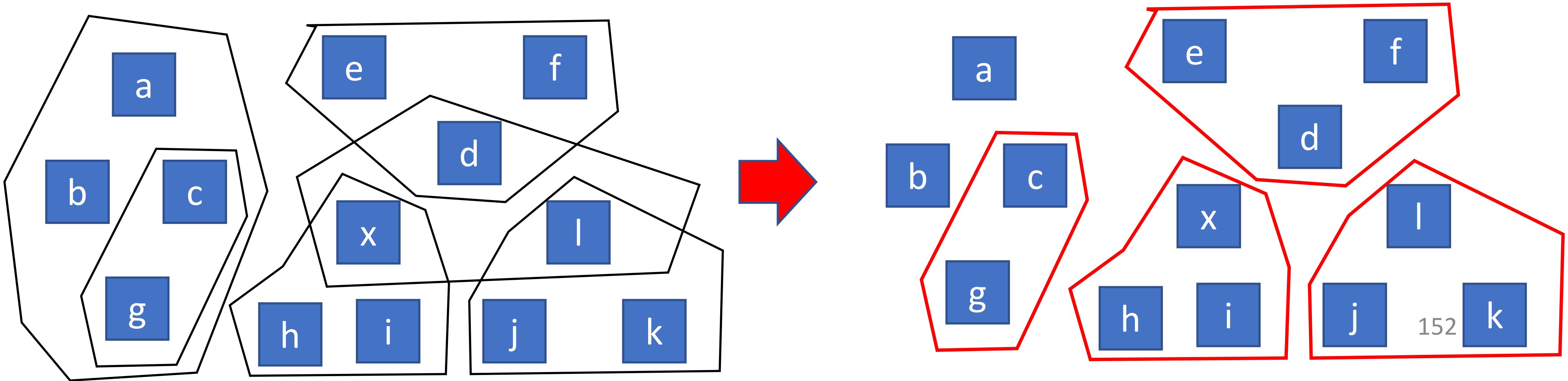
Remark

- In the following, we first show that
  MDCRSP can be reduced to *the set packing problem*

- Our algorithm is obtained by modifying the reduction

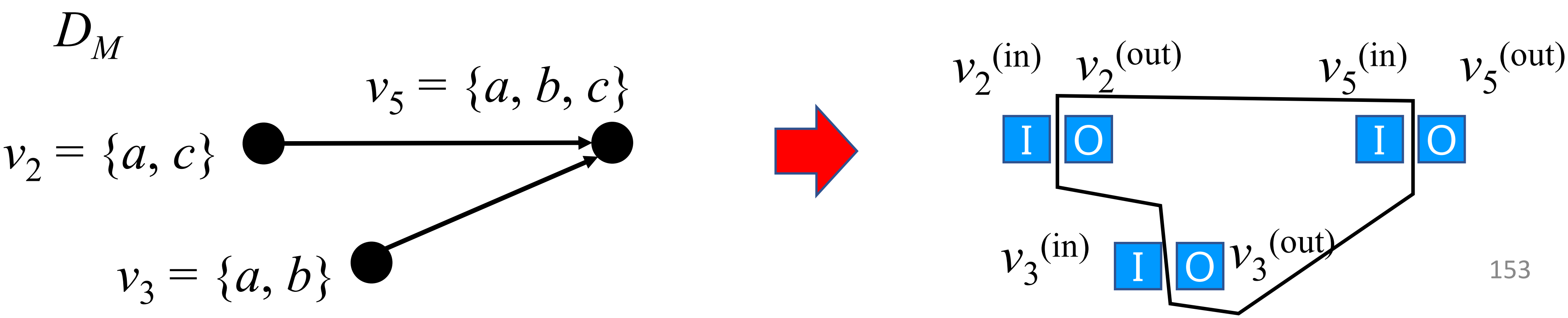
Definition

- The Set Packing problem (SP):
  Given a universe of elements E and
  a family $\mathcal{F}$ of subsets of E,
  find a maximum size subfamily of $\mathcal{F}$ of pairwise disjoint sets



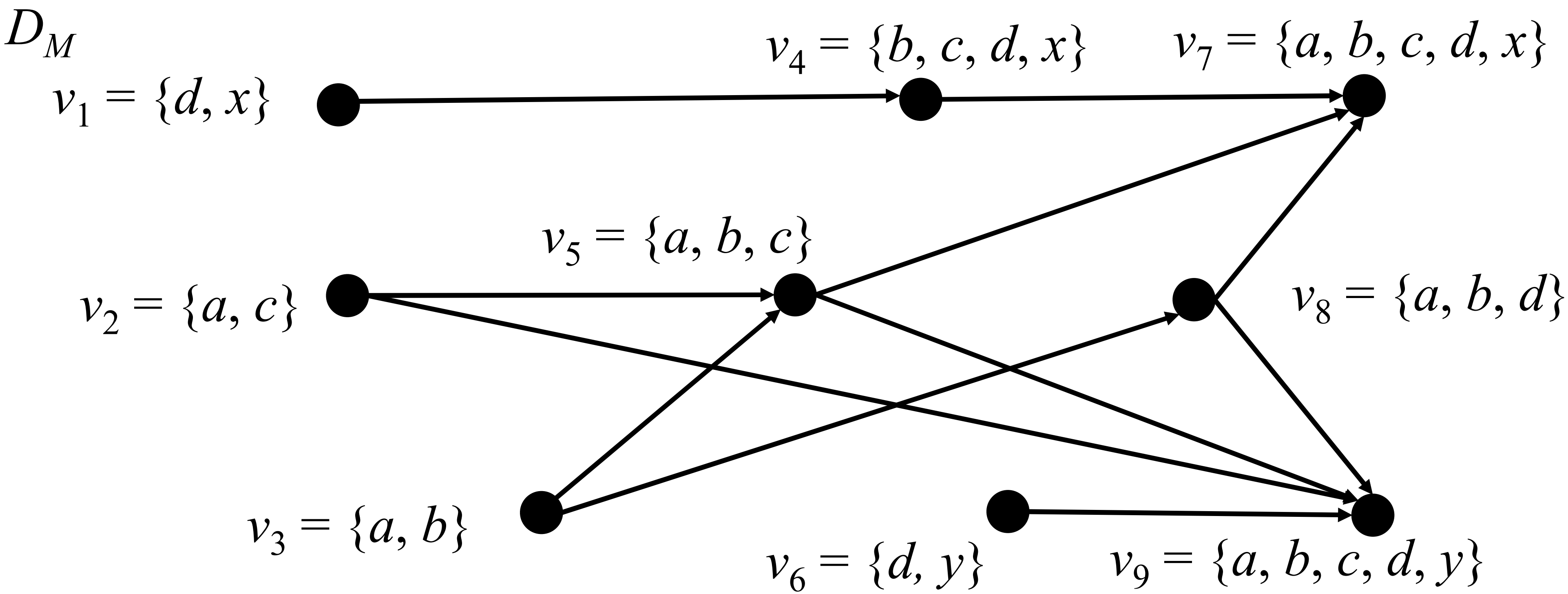
The reduction

- Given an $m \times n$ matrix M, we construct an instance $(E, \mathcal{F})$ of SP as follows
- Each vertex v of D_M is associated with two elements $v^{(\text{in})}$, $v^{(\text{out})}$ in E
- Each candidate (p, Q) is associated with a set $\{p^{(\text{in})}\} \cup Q^{(\text{out})}$, where $Q^{(\text{out})} = \{v^{(\text{out})} \mid v \in Q\}$



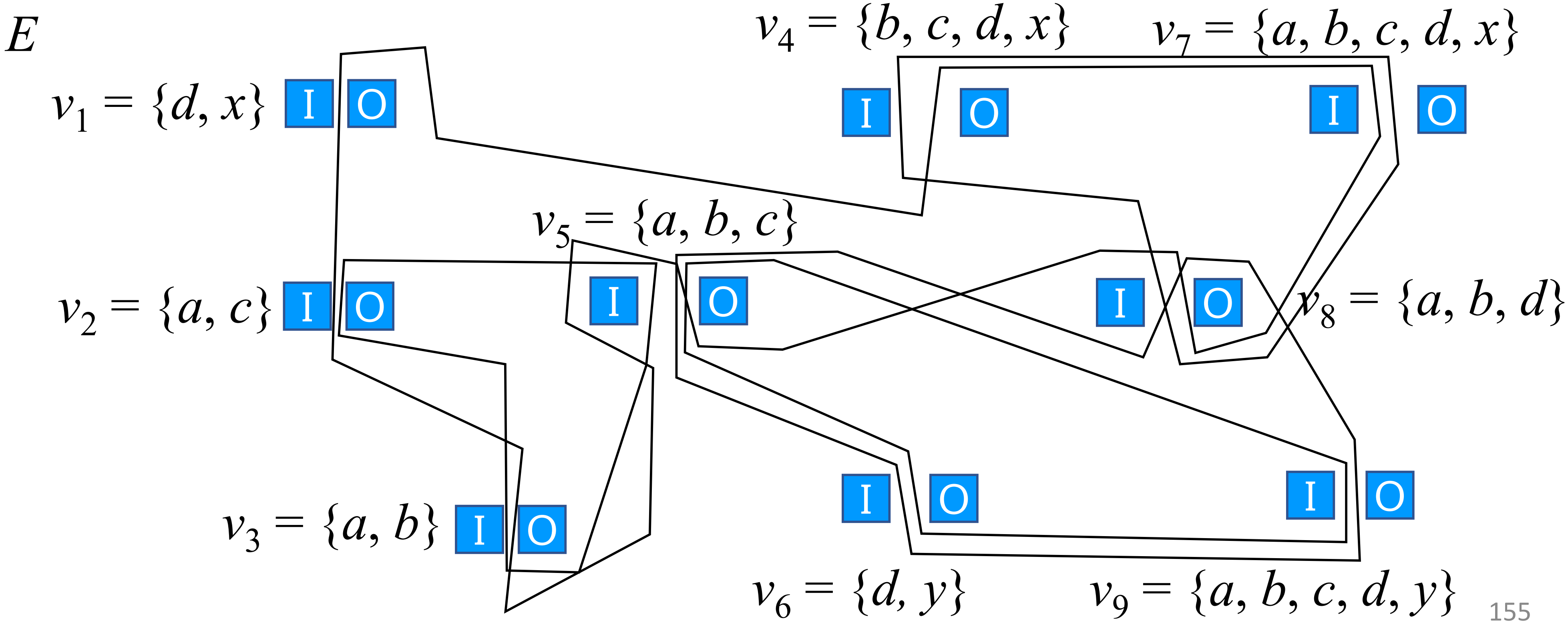
Example

- An example:



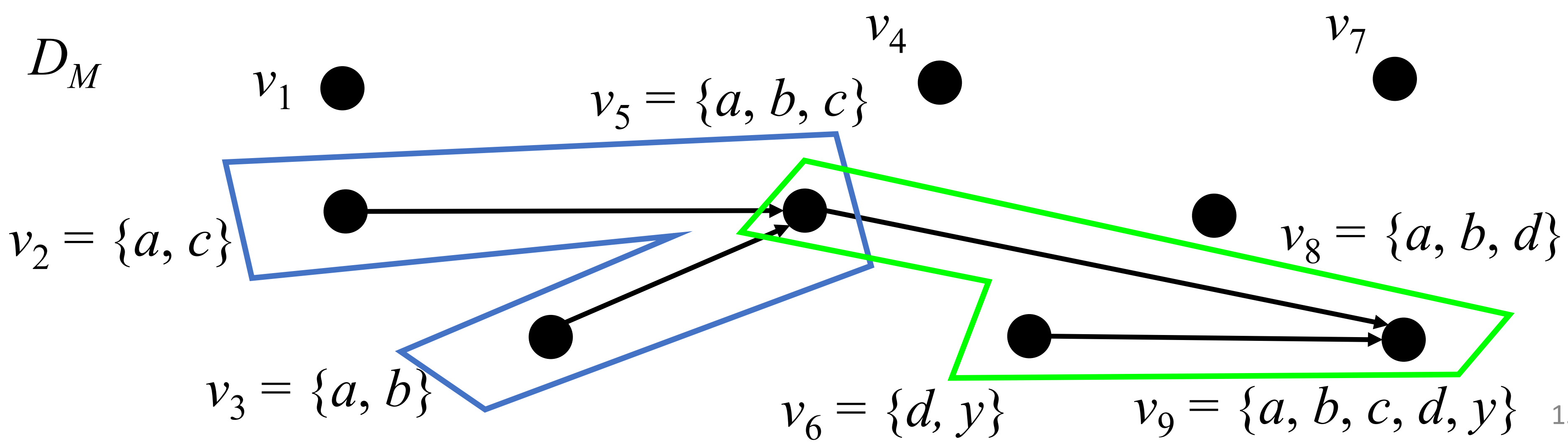
Example

- The universe E
- and some sets (not all sets) in $\mathcal{F}$



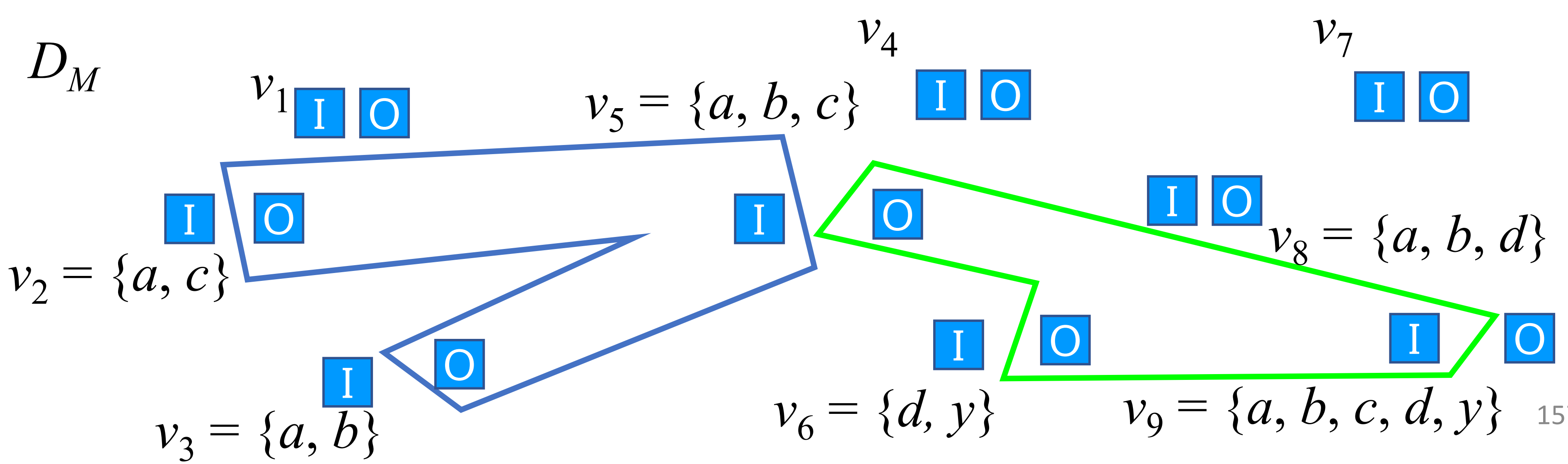
Observation

- Let $C_1 = (p_1, Q_1)$ and $C_2 = (p_2, Q_2)$ be two candidates
- Let S_1, S_2 be the corresponding set of C_1, C_2 in $\mathcal{F}$



Observation

- Let $C_1 = (p_1, Q_1)$ and $C_2 = (p_2, Q_2)$ be two candidates
- Let S_1, S_2 be the corresponding set of C_1, C_2 in $\mathcal{F}$
- Observe that C_1 and C_2 are disjoint
  $\leftrightarrow S_1, S_2$ are disjoint



The reduction

- A *packing* of $\mathcal{F}$ is a collection of disjoint sets in $\mathcal{F}$

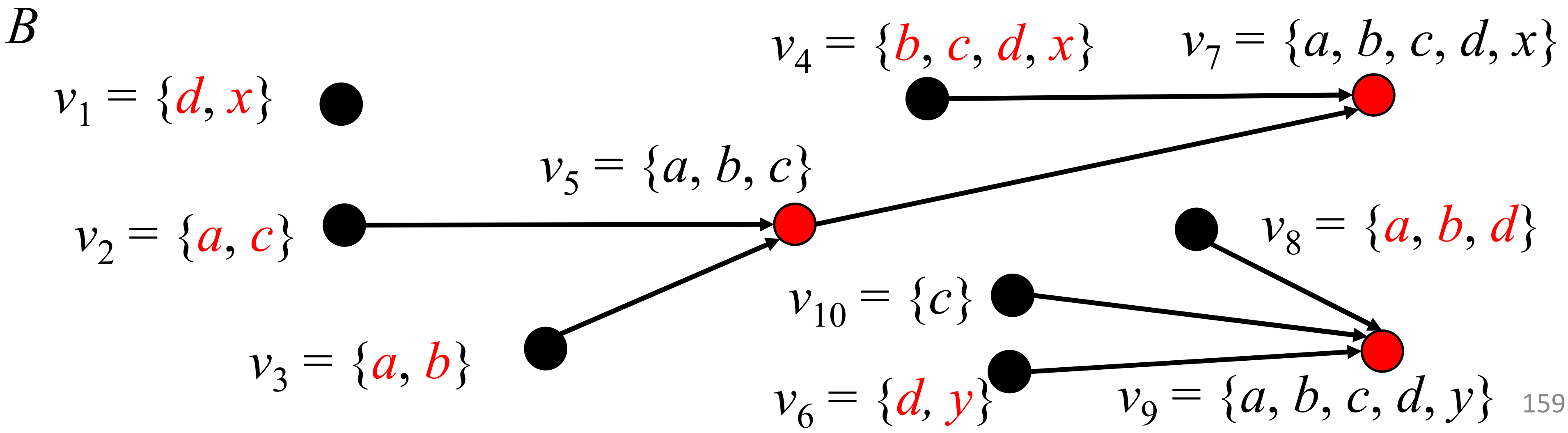
- **Lemma 5.2.** For any matrix M, the following hold:

1. Any trimmed branching B of D_M can be transformed to a packing of $\mathcal{F}$ with size $n - |I(B)|$
2. Any packing $\mathcal{P}$ of $\mathcal{F}$ can be transformed to a trimmed branching B with $|I(B)| = n - |\mathcal{P}|$

Proof

Part 1. trimmed branching $\rightarrow$ packing

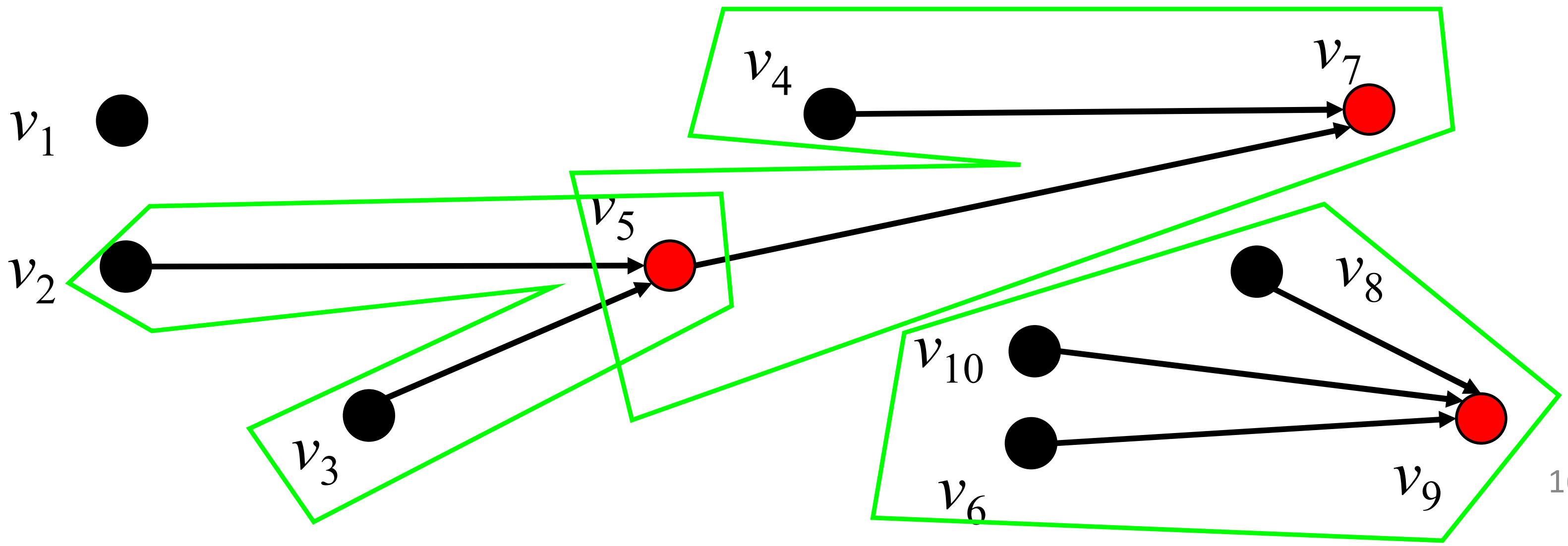
- Let B be a trimmed branching
- Let $p_1, p_2, \ldots, p_x$ be the B-reducible vertices of B, where $x = n - |I(B)|$
- Let Q_i be the child set of p_i



Proof

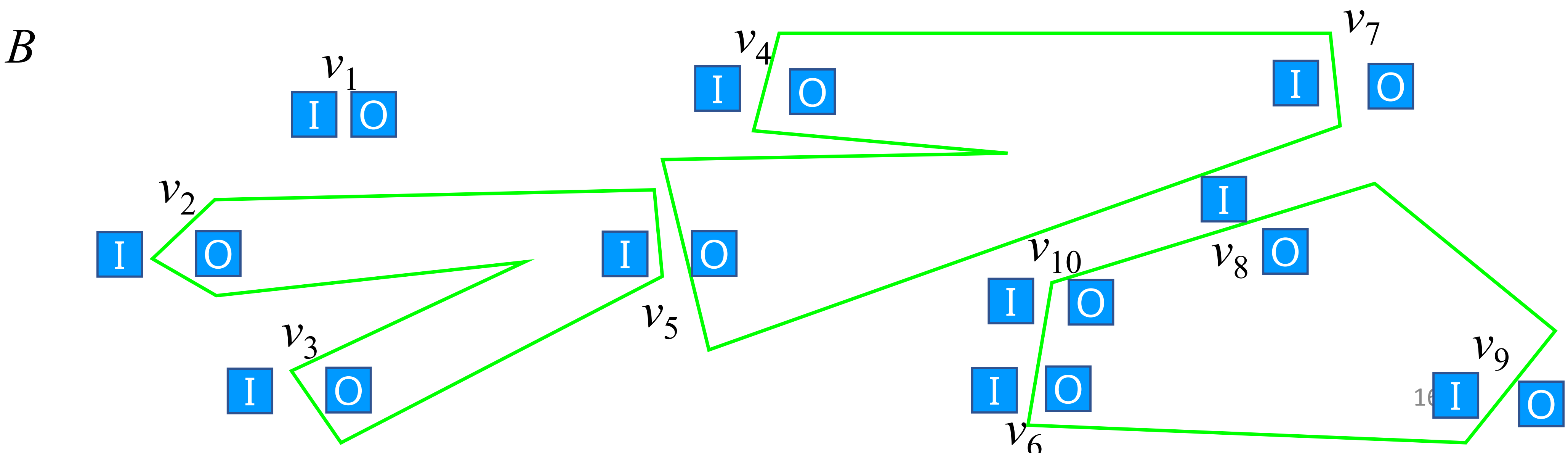
- For $i = 1, 2, \ldots, x$, let $C_i = (p_i, Q_i)$ be a candidate
- Since B is a branching, the sets $Q_1, Q_2, \ldots, Q_x$ are pairwise disjoint
- Thus, $C_1, C_2, \ldots, C_x$ are pairwise disjoint

B



Proof

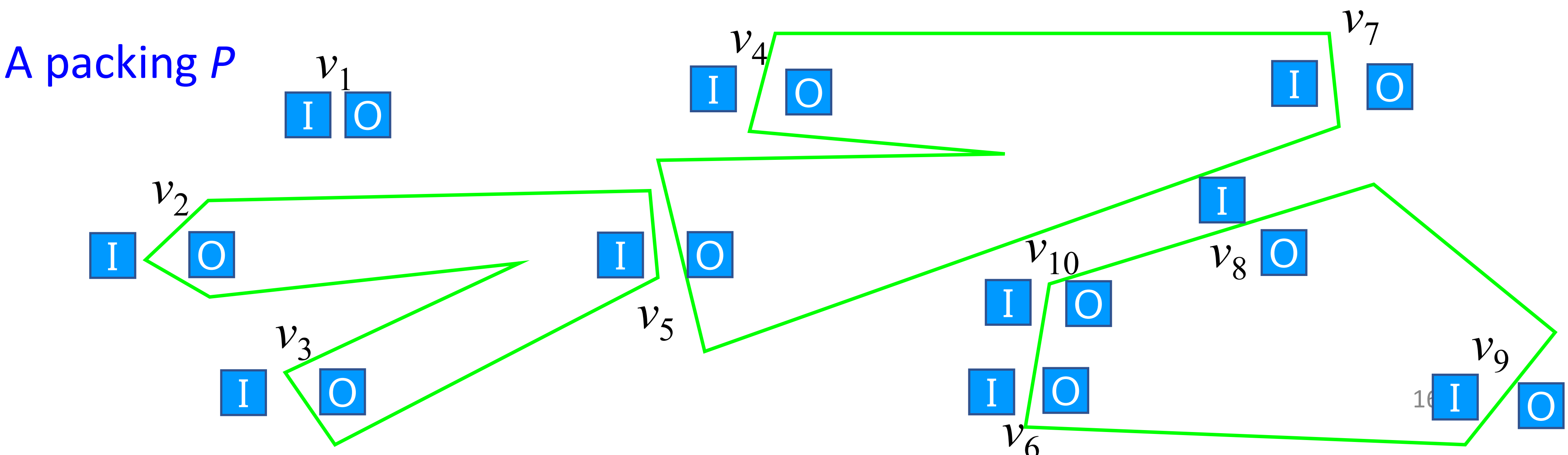
- $C_1, C_2, \ldots, C_x$ corresponds to a subfamily $\{S_1, S_2, \ldots, S_x\}$ of $\mathcal{F}$
- Since $C_1, C_2, \ldots, C_x$ are pairwise disjoint,
  $\{S_1, S_2, \ldots, S_x\}$ is a packing of size $x = n - |I(B)|$



Proof

Part 2. packing $\rightarrow$ trimmed branching

- This part is symmetric to Part 1
- This completes the proof



Remark

- Lemma 5.2 shows that
  the optimal packing of $\mathcal{F}$ has size $n - \zeta(M)$

- This suggests us to use approximation algorithms for SP
  to approximate $n - \zeta(M)$

Remark

- However, there are two difficulties in approximating $\zeta(M)$

  (1) There is no constant approximation algo. for SP
  (unless P = NP)

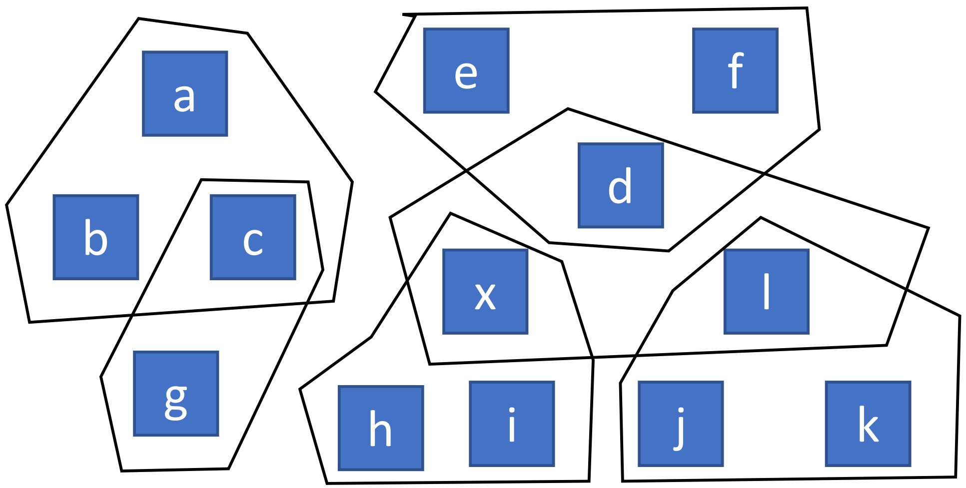
  (2) a constant approximation of $n - \zeta(M)$
  does not imply a constant approximation of $\zeta(M)$

- Because of (1), we will use approximation algorithms for
  the k-Set Packing Problem

Definition

- For a constant k, the *k-Set Packing problem* is SP with an additional constraint:

  each set in the input family $\mathcal{F}$ has size at most k



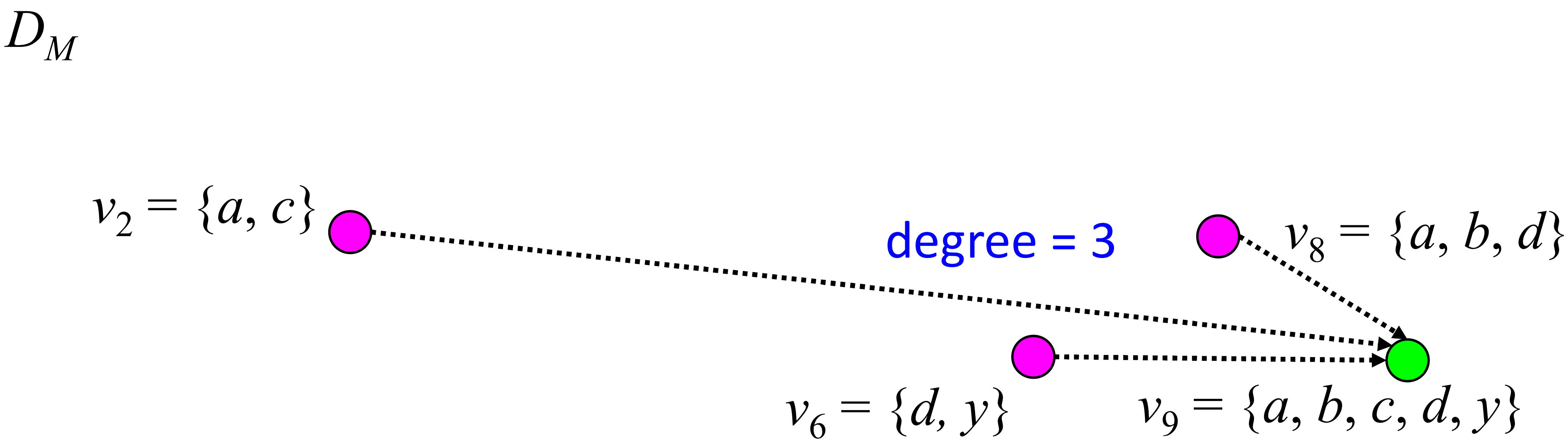
3-SP → size ≤ 3

Remark

- We will present two algorithms
- Both are based on approximation algorithms for *k*-SP
- The first one is efficient and guarantees a ratio of 5/3
- The second one is less efficient but guarantees a ratio of $4/3 + \delta$

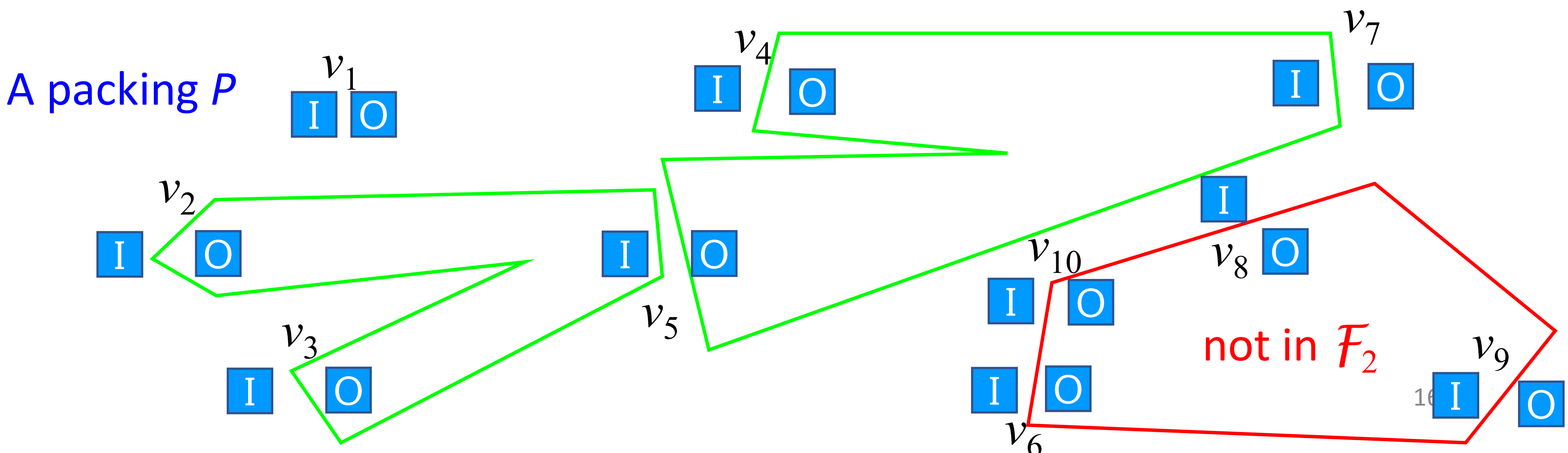
Definition

- The *degree* of a candidate (p, Q) is the size of Q
- Since the vertices of D_M (i.e. the supports of M) are distinct, each candidate has degree ≥ 2



Definition

- For a fixed integer d, let $\mathcal{F}_d$ denote the subfamily of $\mathcal{F}$ which contains all sets in $\mathcal{F}$ with size at most $d + 1$
- That is, $\mathcal{F}_d$ corresponds to the set of candidates of degree at most d
- $(E, \mathcal{F}_d)$ is an instance of $(d + 1)$-SP

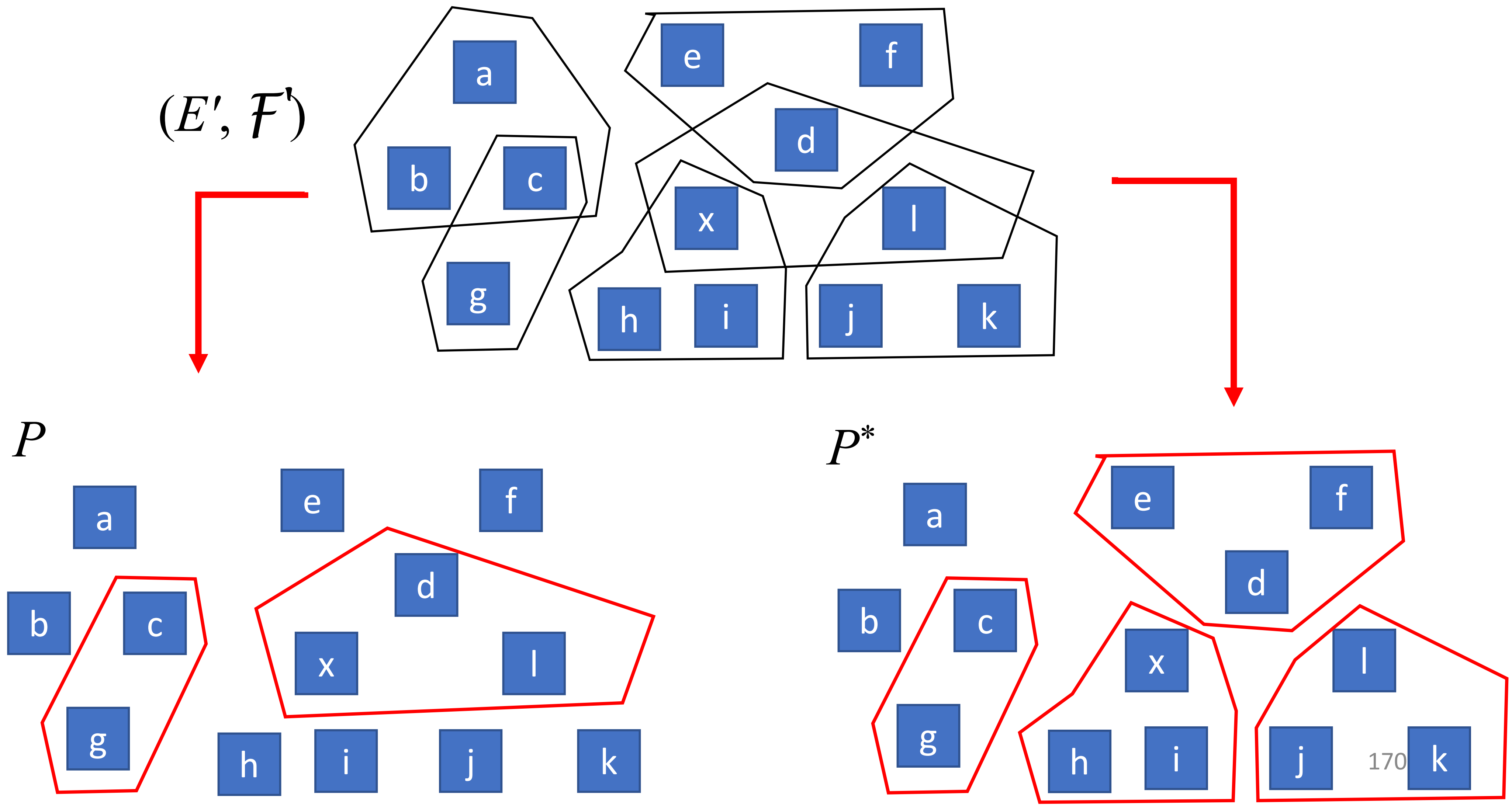


Lemma

- Our first algorithm is based on the following folklore lemma

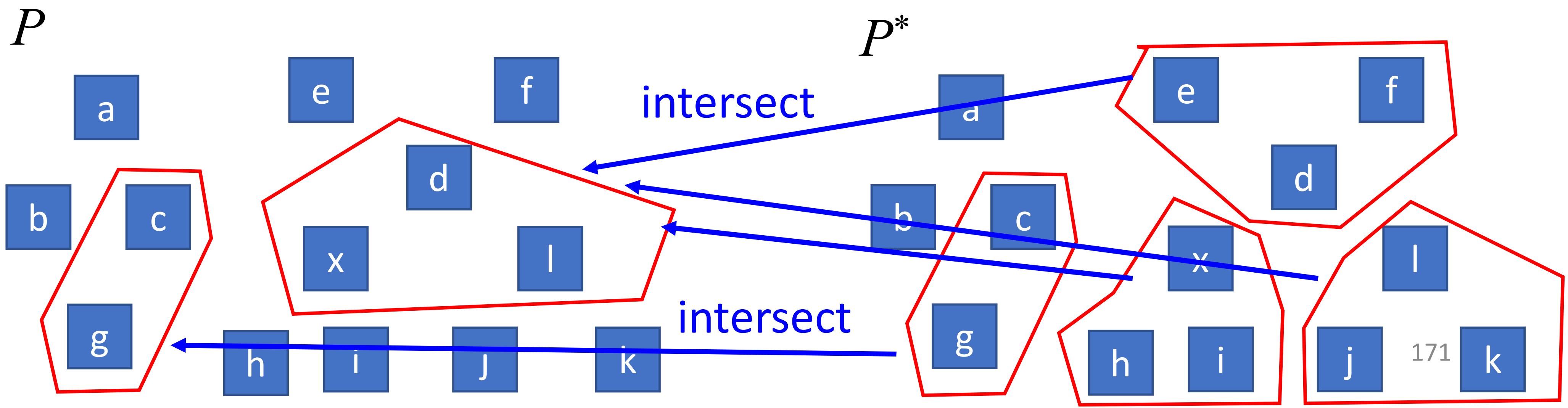
- **Lemma 5.3.** [??] There is a linear time 3-approximation algorithm for 3-SP

- *Proof*. Let $(E', \mathcal{F}')$: an instance of 3-SP
  P^*: the optimal packing of $\mathcal{F}'$
  P: an inclusion-wise maximal packing



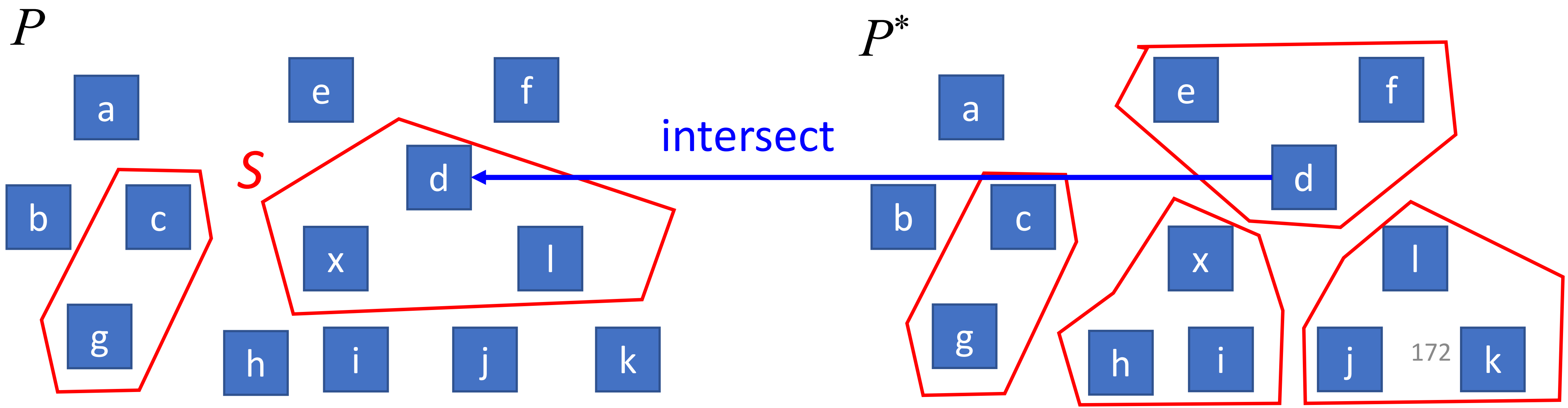
Proof

- We claim that $|P| \geq |P^*| / 3$
- First, since P is maximal,
  each set in P^* intersects at least one set in P



Proof

- Consider a set S in P
- Since P^* is a packing,
  each element in S is in at most one set in P^*
- Since the size of S is at most 3,
  S intersects at most 3 sets in P^*



Proof

- In summary:

  - each set in P^* intersects a least one set in P

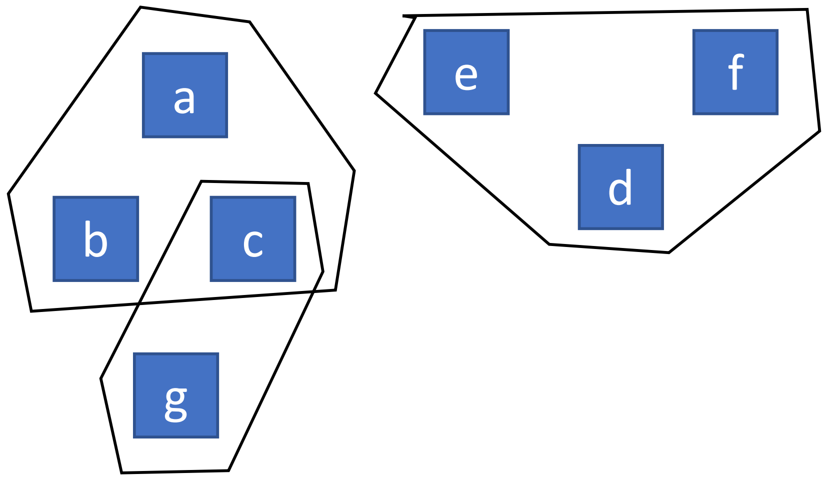
  - each set in P intersects at most 3 sets in P^*
    $\rightarrow$ P has at least $|P^*| / 3$ sets

- Thus, it suffices to show that
  a maximal packing can be found in linear time

Proof

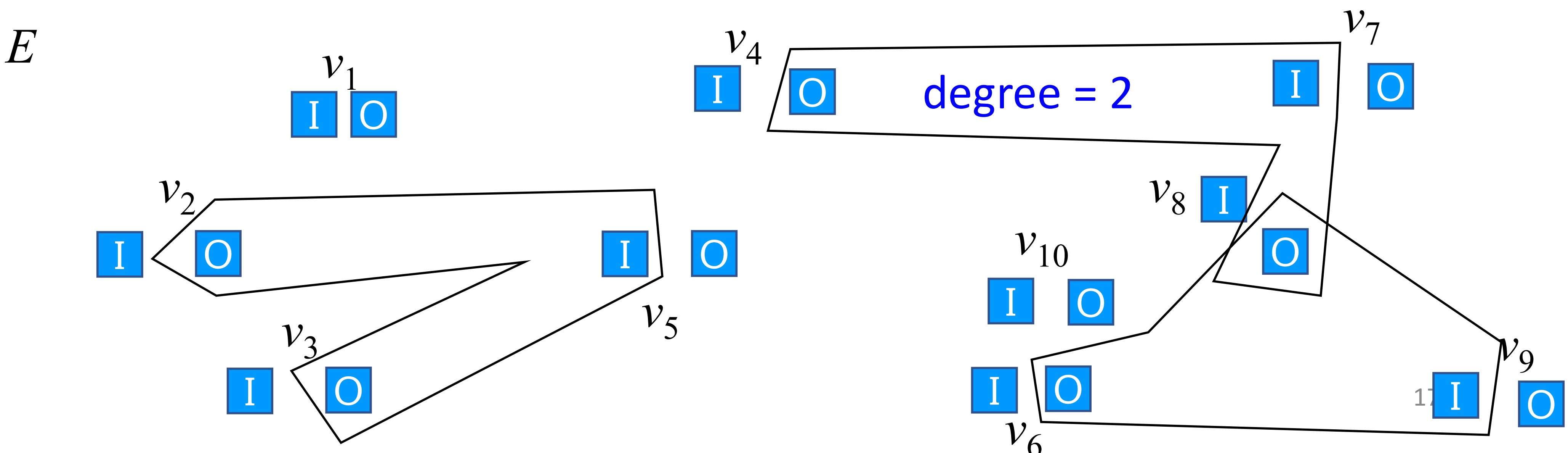
- Maintain a packing P'; Initially, set $P' = \varnothing$
- For each set S in $\mathcal{F}'$:

  if $P' \cup \{S\}$ is a packing, set $P' = P' \cup \{S\}$
- The checking can be done in $O(3) = O(1)$ time
- This completes the proof



Algorithm 1

- Our first algorithm is as follows
  - Step 1. compute D_M
  - Step 2. compute $(E, \mathcal{F}_2)$
  - Step 3. find a packing P of $\mathcal{F}_2$ by using Lemma 5.3
  - Step 4. transform P to a branching B
  - Step 5. output B



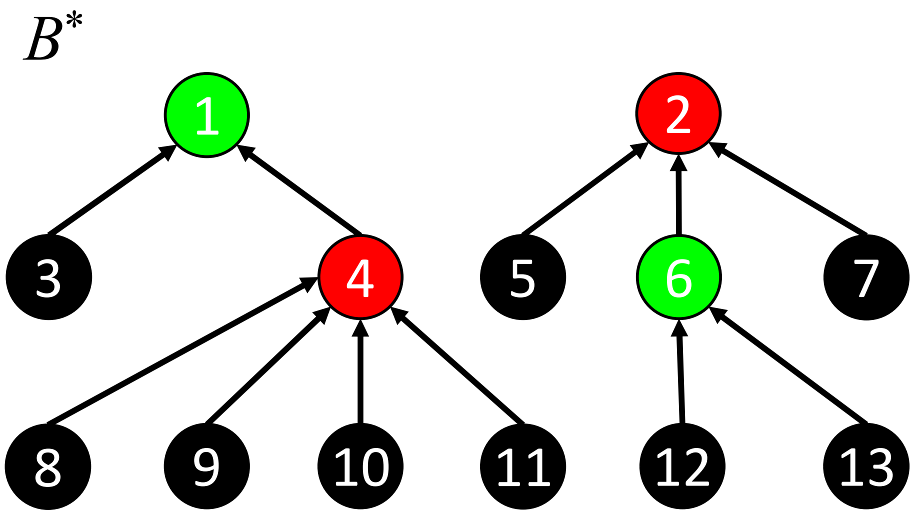
Theorem

- **Theorem 5.4.** Algorithm 1 is a (5/3)-approximation algorithm for MDCRSP with time complexity $O(mn^2)$

- *Proof*. We first analyze the approximation ratio

Proof

- Let
  - B^*: an optimal trimmed branching of D_M
  - X_2 : the set of B^*-reducible vertices with in-degree 2
  - X_3 : the set of B^*-reducible vertices with in-degree ≥ 3
- Note that $\{X_2, X_3\}$ is a partition of B^*-reducible vertices



Proof

- Since B^* has $|X_2|$ reducible vertices of degree 2,
  $\mathcal{F}_2$ has a packing of size $|X_2|$

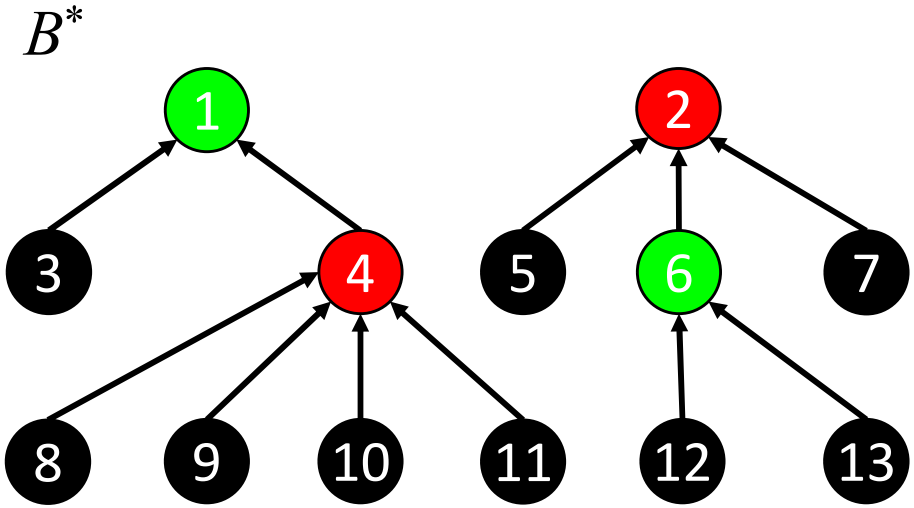
- (Recall that $\mathcal{F}_2$ corresponds to the candidates of degree 2)

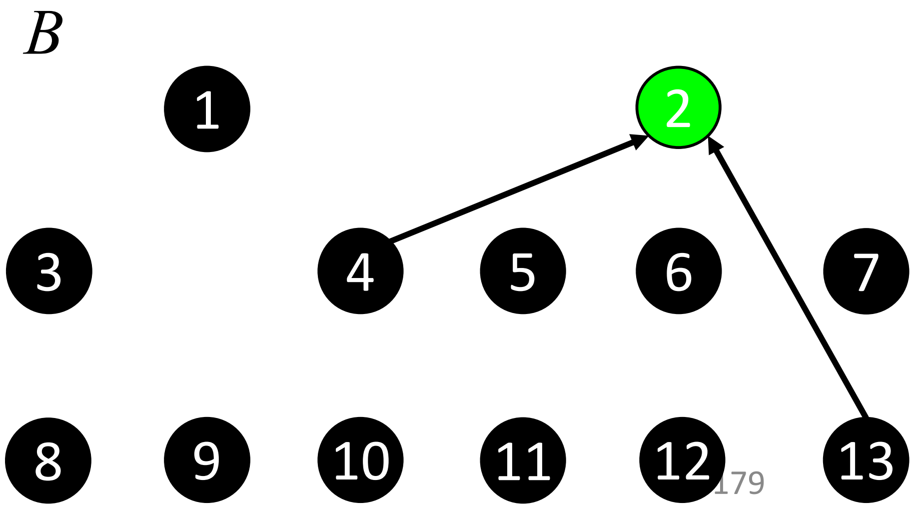
- Recall that the packing P is obtained by performing
  a 3-approximation algorithm on $\mathcal{F}_2$

- Thus, $|P| \geq |X_2| / 3$

Proof

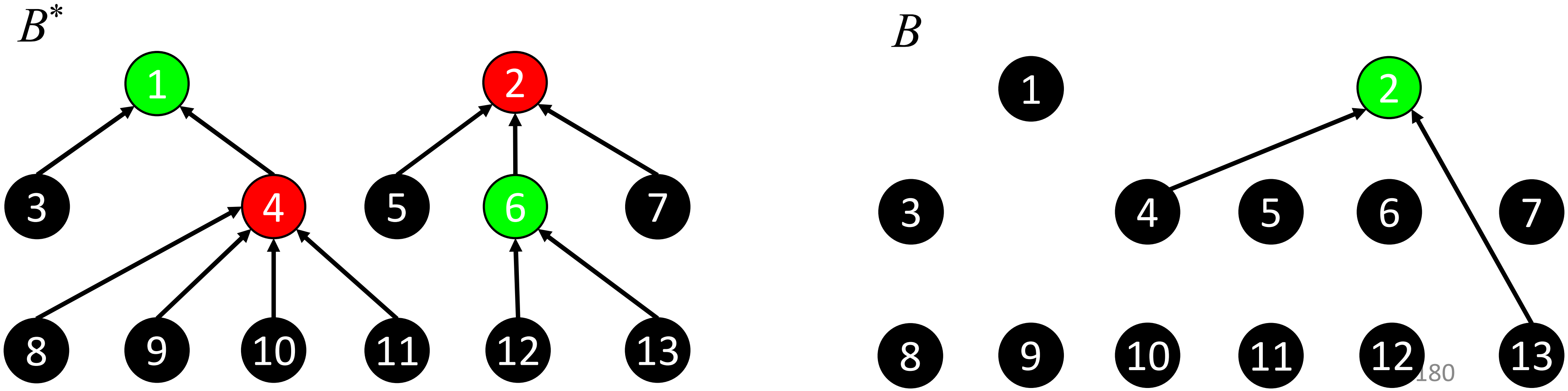
- Since B is the corresponding branching of P,
  B has at least $|X_2|$ / 3 reducible vertices (of degree 2)





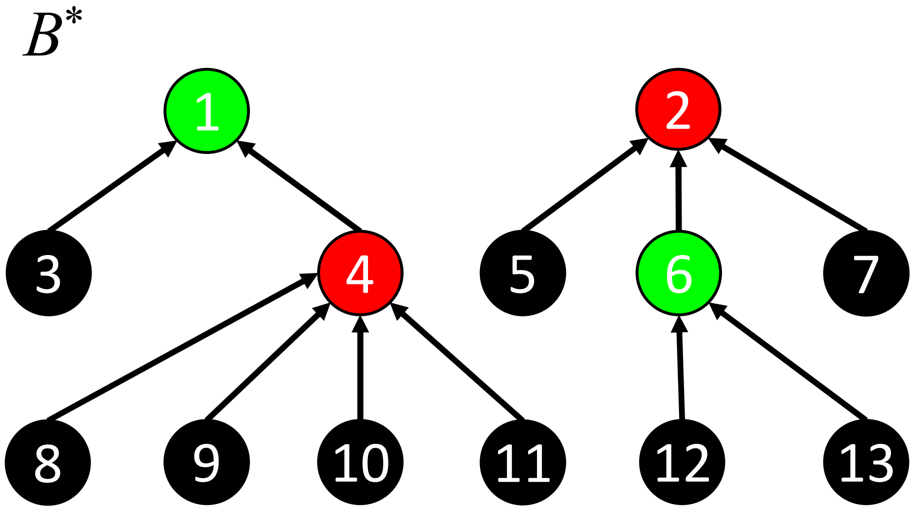
Proof

- The cost of B^*: $n - |X_2| - |X_3|$
- The cost of B: $\leq n - |X_2| / 3$
- Goal: upper bound the ratio $(n - |X_2| / 3) / (n - |X_2| - |X_3|)$



Proof

- Since B^* is a branching, (the sum of in-degrees) $\leq n - 1$
- Thus, we have $2|X_2| + 3|X_3| \leq n - 1$



Remark

- We claim that the ratio $(n - |X_2| / 3) / (n - |X_2| - |X_3|) \leq 5/3$

- Before proceeding the proof,
  we present a rough estimation of
  the worst-case performance of Algorithm 1

Estimation

- In summary:
  - $|X_2|$ = (# reducible vertices with in-degree 2 in B^*);
  - $|X_3|$ = (# reducible vertices with in-degree > 2 in B^*)
  - B has at least $|X_2| / 3$ reducible vertices
  - $2|X_2| + 3|X_3| \leq n - 1$

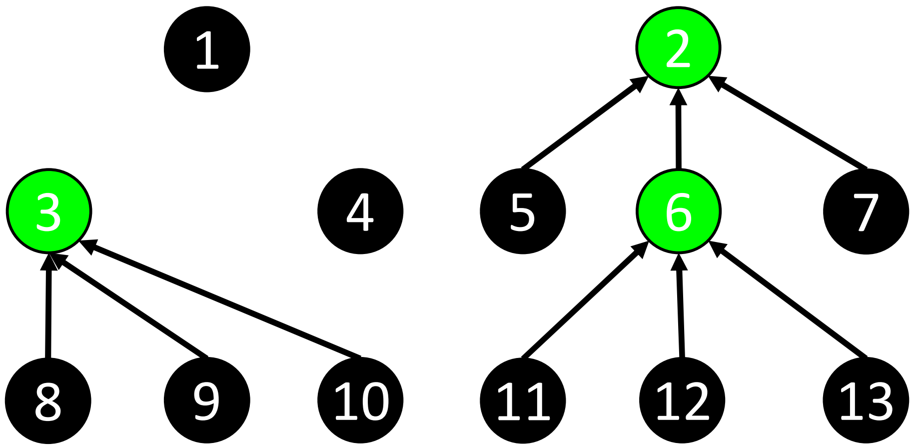
Estimation

- Consider two bad cases for our algorithm
- **Bad case 1**: in B^*, $|X_2| = 0$, $|X_3| \approx n / 3$
  - The cost of $B^* \approx n - n / 3 = (2/3)n$
  - The cost of $B \leq n - |X_2| / 3 = n - 0$

$\rightarrow$ in this case, the ratio $= n / ((2/3)n) = 3/2 < 5/3$

(our goal)

B^*

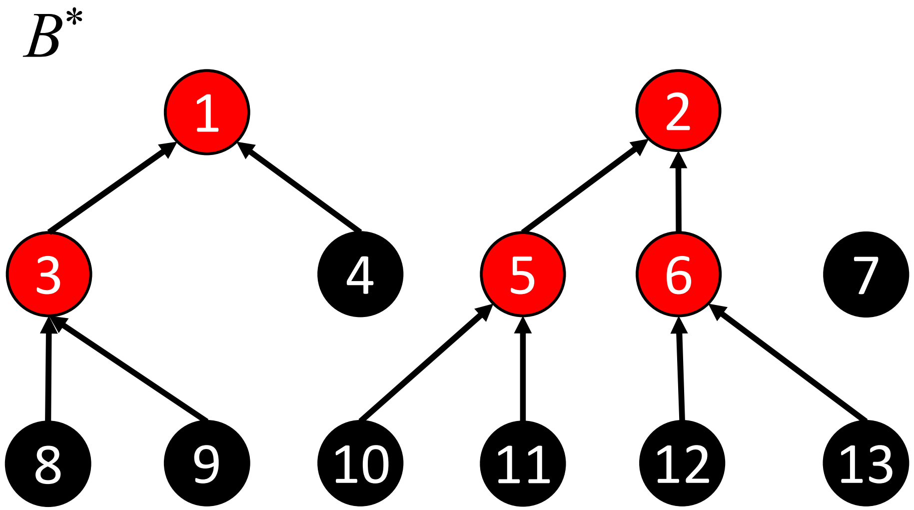


Estimation

- **Bad case 2**: in B^*, $|X_2| \approx n / 2$, $|X_3| = 0$
  - The cost of $B^* \approx n - n / 2 = (1/2)n$
  - The cost of $B \leq n - |X_2| / 3 \approx (5/6)n$

    $\rightarrow$ in this case, ratio = $(5/6)n / (1/2)n = 5 / 3$

    (our goal)
- It turns out that this is the worst case



Back to the proof

- Let $\alpha = |X_2| / n$ and $\beta = |X_3| / n$
- α is the portion of
  vertices which are reducible and have in-degree 2
- Note that α and β are non-negative

Back to the proof

- Our analysis is rephrased as follows:

  - ratio $\leq (n - |X_2| / 3) / (n - |X_2| - |X_3|)$
    $= (1 - \alpha / 3) / (1 - \alpha - \beta)$ $\quad (/\ n)$
    $= (3 - \alpha) / (3 - 3\alpha - 3\beta)$ $\quad (\times\ 3)$

  - $2|X_2| + 3|X_3| \leq n - 1$
    $\rightarrow\ 2\alpha + 3\beta \leq (n - 1) / n \leq 1$

  - $\alpha, \beta \geq 0$

Proof

- To sum up, we have the following mathematical program:

$$\text{Program 1: maximize } \frac{3-x}{3-3x-3y}$$
$$\text{subject to (I1) } 2x + 3y \le 1 \text{ and}$$
$$\text{(I2) } x, y \ge 0$$

- Let r^* be the maximum objective value of Program 1

- Since (α, β) is a feasible point of Program 1,
the approximation ratio is upper bounded by r^*

Proof

> Program 1: maximize $\dfrac{3 - x}{3 - 3x - 3y}$
> subject to (I1) $2x + 3y \leq 1$ and
> (I2) $x, y \geq 0$

- By (I1), the objective value is always positive
- Thus, the objective value increases as y increases
- Hence, there is a maximizer (x^*, y^*) with $2x^* + 3y^* = 1$,
  or equivalently, $y^* = (1 - 2x^*) / 3$
- Therefore, we can rewrite Program 1 with $y = (1 - 2x) / 3$

Proof

Program 1: maximize $\dfrac{3-x}{3-3x-3y}$

subject to $2x+3y=1$ and

$x, y \geq 0$

- By $y = (1-2x)/3$,
  the objective function is rewritten as

$$\frac{3-x}{3-3x-(1-2x)} = \frac{3-x}{2-x}$$

Proof

Program 1: maximize $\dfrac{3 - x}{3 - 3x - 3y}$

subject to $2x + 3y = 1$ and

$x, y \geq 0$

is equivalent to

Program 2: maximize $f(x) = \dfrac{3 - x}{2 - x}$

subject to $2x \leq 1$ (and $y = (1 - x) / 3$) and

$x \geq 0$

Proof

Program 2: maximize $f(x) = \dfrac{3 - x}{2 - x}$
subject to $2x \leq 1$ and
$x \geq 0$

- The derivative of $f(x)$ is

Quotient rule

$$\left(\frac{g}{h}\right)' = \frac{g'h - gh'}{h^2}$$

$$f'(x) = \frac{-(2-x)+(3-x)}{(2-x)^2}$$

$$= \frac{1}{(2-x)^2}$$

$$> 0 \text{ for } x \neq 2$$

$\rightarrow$ f is strictly increasing in the range: $0 \leq 2x \leq 1$

Proof

Program 2: maximize $f(x) = (3 - x) / (2 - x)$
subject to $2x \leq 1$ and
$x \geq 0$

- Since $2x \leq 1$, Program 2 is maximized when $x = 0.5$
- The maximum is $f(0.5) = \frac{2.5}{1.5} = \frac{5}{3}$
- Consequently, the algorithm guarantees a ratio of $\frac{5}{3}$

Proof

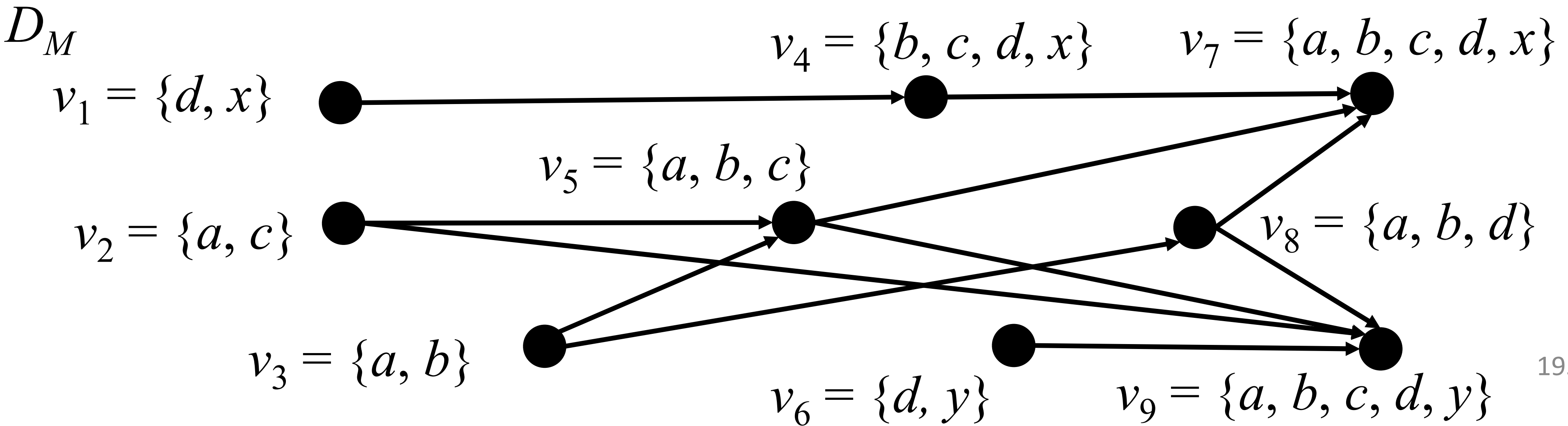
- We proceed to show that Algorithm 1 runs in $O(mn^2)$ time

- Recall Algorithm 1:
  - Step 1. compute D_M
  - Step 2. compute $(E, \mathcal{F}_2)$
  - Step 3. find a packing P of $\mathcal{F}_2$ by using Lemma 5.3
  - Step 4. transform P to a branching B
  - Step 5. output B

Proof

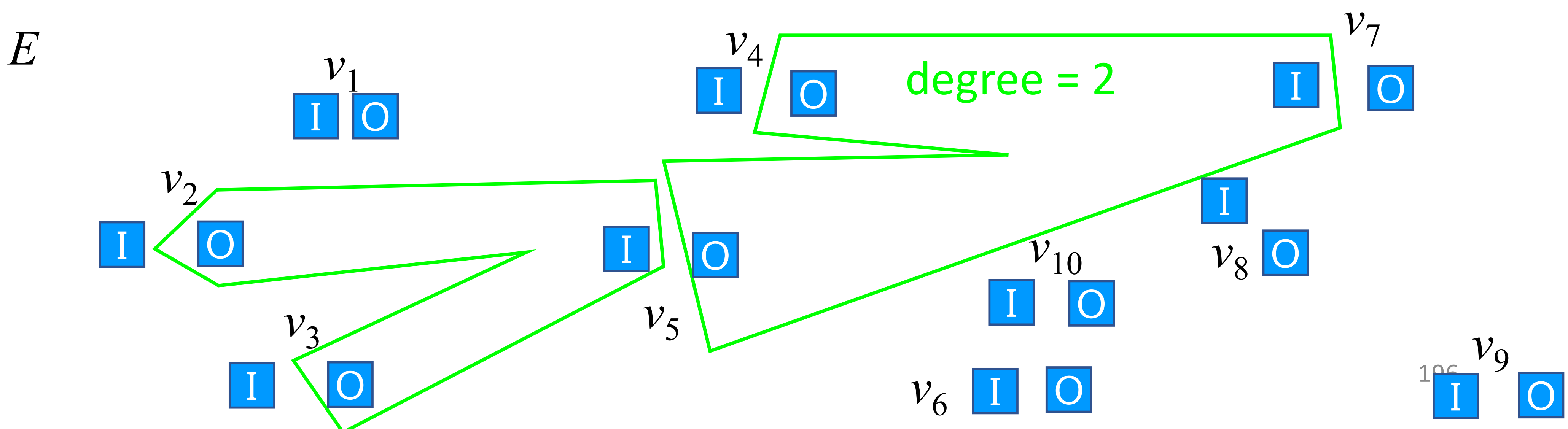
- Step 1. compute D_M

- By my master thesis, D_M can be computed in $O(\max(mn^{1.373}, m^{0.373}n^2)) = O(mn^2)$ time



Proof

- Step 2. compute $(E, \mathcal{F}_2)$
- $E = \{v^{(\text{in})} \mid v \in V(D_M)\} \cup \{v^{(\text{out})} \mid v \in V(D_M)\}$
  can be obtained in $O(n)$ time
- Recall that $\mathcal{F}_2$ corresponds to the candidates
  $\{(p, Q\} \mid Q$ contains at most two vertices$\}$
  $= \{(p, Q\} \mid Q$ contains exactly two vertices$\}$



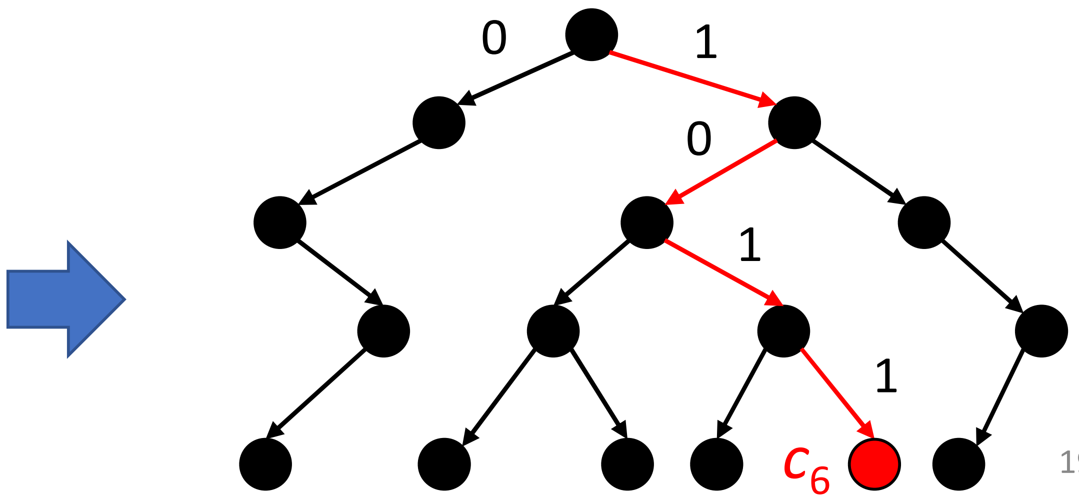
Proof

- For each pair of vertices u, v (u, v containment?) $\in V(D_M)$:
  - Let $p = u \cup v$
  - If $p \in V(D_M)$, $p \neq u$ and $p \neq v$,
    add $\{p^{(\text{in})}, u^{(\text{out})}, v^{(\text{out})}\}$ to $\mathcal{F}_2$
- We need an efficient data structure to check if $p \in V(D_M)$

Proof

- Recall that each vertex of D_M represents a column of M
- Thus, a vertex (set of rows) can be represented by an *m*-bit vector
- We build a *digital search tree* T on the set of columns
- Each leaf has depth m and represents a column of M

M	c_1	c_2	c_3	c_4	c_5	c_6
a	1	1		1	1	1
b					1	0
c			1	1	1	1
d		1				1

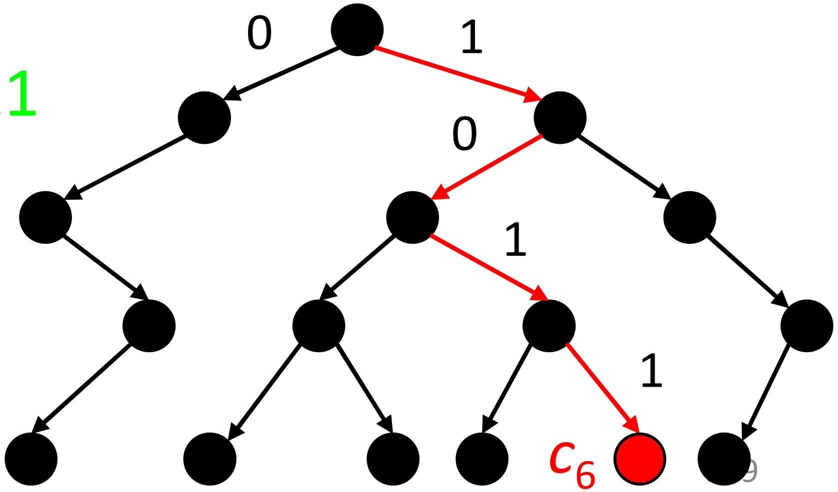


Proof

- Recall: for each pair of vertices $u, v \in V(D_M)$:
  - Let $p = u \cup v$ → $O(m)$ time
  - If $p \in V(D_M)$, $p \neq u$ and $p \neq v$, → $O(m)$ time
    add $\{p^{(\text{in})}, u^{(\text{out})}, v^{(\text{out})}\}$ to $\mathcal{F}_2$
    → Step 2 takes $O(mn^2)$ time

M	c_1	c_2	c_3	c_4	c_5	c_6
a	1	1		1	1	1
b					1	
c			1	1	1	1
d		1				1

$c_2 \cup c_4 = 1011$



Proof

- Step 3. find a packing P of $\mathcal{F}_2$ by using Lemma 5.3
- Recall that the algorithm in Lemma 5.3 takes linear time
- Since $|\mathcal{F}_2| = O(n^2)$, Step 3 takes $O(n^2)$ time

Proof

- Step 4. transform P to a branching B
- Step 5. output B

- Clearly, Step 4 and 5 can be done in $O(n)$ time

- Since each step is done in $O(mn^2)$ time,
  the proof is complete

Algorithm 2

- We proceed to present Algorithm 2

- It is based on the result of [A]

- **Theorem 5.5.** [A] There is a deterministic algorithm for *k*-SP which, given any $\varepsilon > 0$, achieves an approximation ratio $\frac{k+1}{3} + \varepsilon$ in time $n^{O(k^3 / \varepsilon^2)}$.

Theorem

- **Theorem 5.6.** For any $\delta > 0$, there is a polynomial time $\left(\frac{4}{3} + \delta\right)$-approximation algorithm for MDCRSP

- *Proof*. Let δ be a fixed positive real number

- W.L.O.G., assume that $\delta < 1$

- We first use Theorem 5.5 to obtain algorithms for 3-SP and 4-SP

Proof

Theorem 5.5.
$\left(\frac{k+1}{3}+\varepsilon\right)$-approximation for k-SP

- Let $\rho_1 = \frac{4}{3-2\delta} > \frac{4}{3}$
- Since $\rho_1 > \frac{4}{3}$, by Theorem 5.5, there exists a ρ_1-approximation algorithm A_1 for 3-SP
- Let $a_1 = \frac{1}{\rho_1} = \frac{3-2\delta}{4} = \frac{3}{4} - \frac{1}{2}\delta$
- By definition, given an instance of 3-SP with optimal packing size t, A_1 finds a packing with size $\geq t / \rho_1 = a_1 \cdot t$

Proof

Theorem 5.5.
$\left(\frac{k+1}{3}+\varepsilon\right)$-approximation for k-SP

- Let $\rho_2 = \frac{10}{6-5\delta} > \frac{10}{6} = \frac{5}{3}$

- Again, by Theorem 5.5, there exists
  a ρ_2-approximation algorithm A_2 for 4-SP

- Let $a_2 = \frac{1}{\rho_2} = \frac{6-5\delta}{10} = \frac{3}{5} - \frac{1}{2}\delta$

- By definition, given an instance of 4-SP with
  optimal packing size t,
  A_1 finds a packing with size $\geq t\ /\ \rho_2 = a_2 \cdot t$

Algorithm 2

- Input: a matrix M and a positive real number δ
- Output: a branching B of D_M
  - Step 1. compute D_M, E, $\mathcal{F}_2$ and $\mathcal{F}_3$
  - Step 2. run A_1 on $\mathcal{F}_2$ to obtain a packing P_1
    run A_2 on $\mathcal{F}_3$ to obtain a packing P_2
  - Step 3. transform P_1 to a branching B_1,
    transform P_2 to a branching B_2
  - Step 4. output the branching with smaller cost

Proof

- Clearly, Algorithm 2 runs in polynomial time

- We will show that

  B costs no more than $\left(\frac{4}{3} + \delta\right)$ times the optimal cost

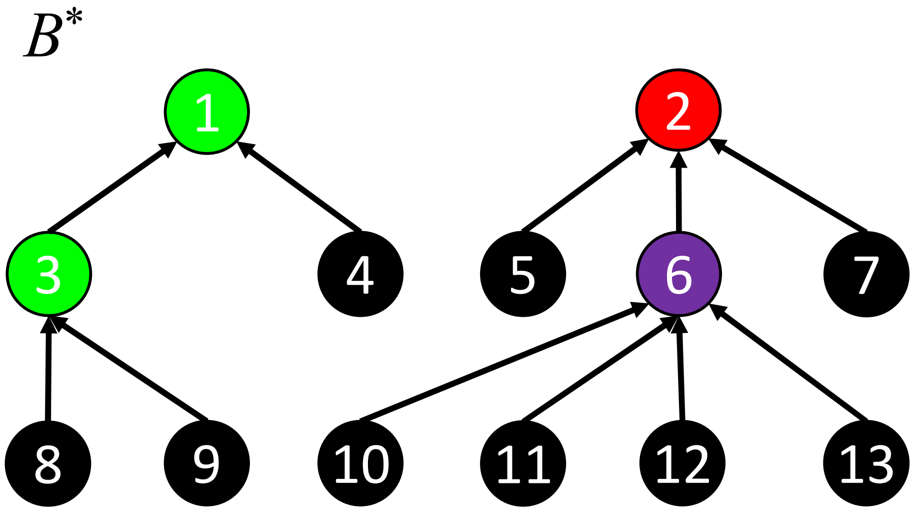
- Interestingly,

  each of B_1 and B_2 does not have this property

Proof

- Let
  - B^*: an optimal branching of D_M
  - Y_2 : the set of reducible vertices with in-degree 2
  - Y_3 : the set of reducible vertices with in-degree 3
  - Y_4 : the set of reducible vertices with in-degree ≥ 4

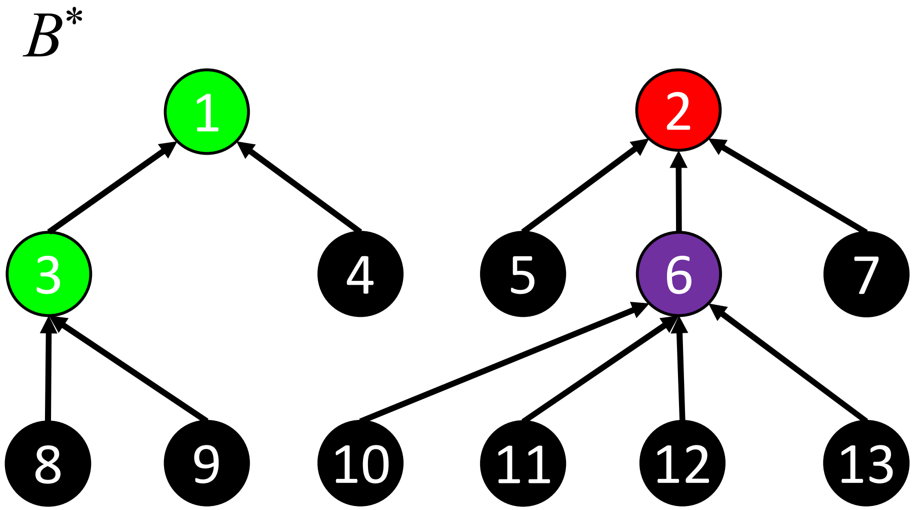
- Y_2, Y_3 and Y_4 patition the set of reducible vertices

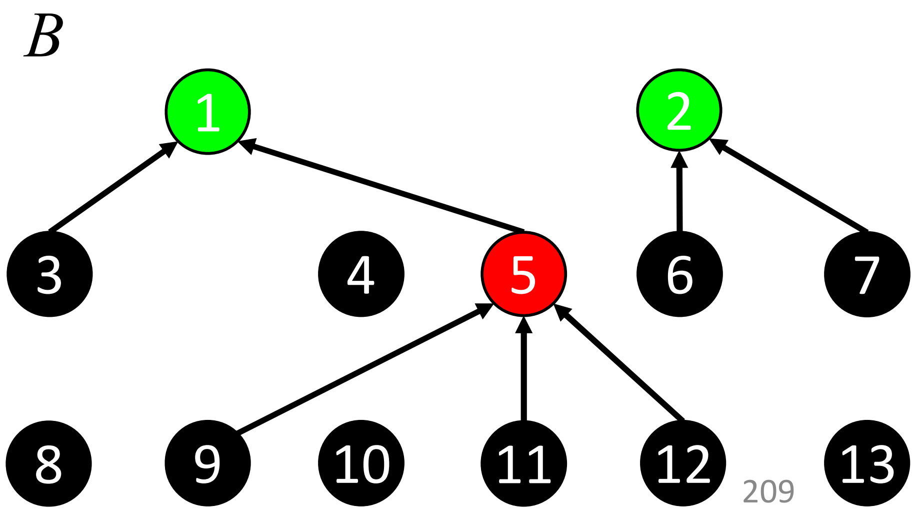


Proof

- Let B be the output of Algorithm 2

- Similar to Theorem 5.4, we upper bound the cost of B in terms of $|Y_2|$ and $|Y_3|$





Proof

- Recall:
  - $\mathcal{F}_2$ corresponds to the set of candidates with degree ≤ 2
  - $\mathcal{F}_3$ corresponds to the set of candidates with degree ≤ 3

- B^* has:
  - $|Y_2|$ reducible vertices of in-degree 2
  - $|Y_3|$ reducible vertices of in-degree 3

- As a result, $\mathcal{F}_2$ has a packing of size $|Y_2|$, and
  $\mathcal{F}_3$ has a packing of size $|Y_2| + |Y_3|$

Proof

- In last page: $\mathcal{F}_2$ has a packing of size $|Y_2|$, and
  $\mathcal{F}_3$ has a packing of size $|Y_2| + |Y_3|$

- Recall that the packing P_1 is obtained by running A_1 on $\mathcal{F}_2$,
  $\rightarrow$ the size of P_1 is at least $a_1 \cdot |Y_2|$
  $\rightarrow$ the cost of $B_1 \leq n - a_1 \cdot |Y_2|$

- Similarly, the cost of $B_2 \leq n - a_2 \cdot (|Y_2| + |Y_3|)$

Proof

- Recall that B is
  the branching with smaller cost among B_1 and B_2

- We obtain the following upper bounds on the costs:

$$\begin{aligned}
\text{cost of } B_1 &\leq n - a_1 \cdot |Y_2|; \\
\text{cost of } B_2 &\leq n - a_2 \cdot (|Y_2| + |Y_3|); \\
\text{cost of } B &= \min(\text{cost of } B_1, \text{cost of } B_2) \\
&\leq \min(n - a_1 \cdot |Y_2|,\ n - a_2 \cdot (|Y_2| + |Y_3|)); \\
\text{cost of } B^* &= n - |Y_2| - |Y_3| - |Y_4|.
\end{aligned}$$

Proof

- Thus, (the cost of B) / (the cost of B^*) $\leq$

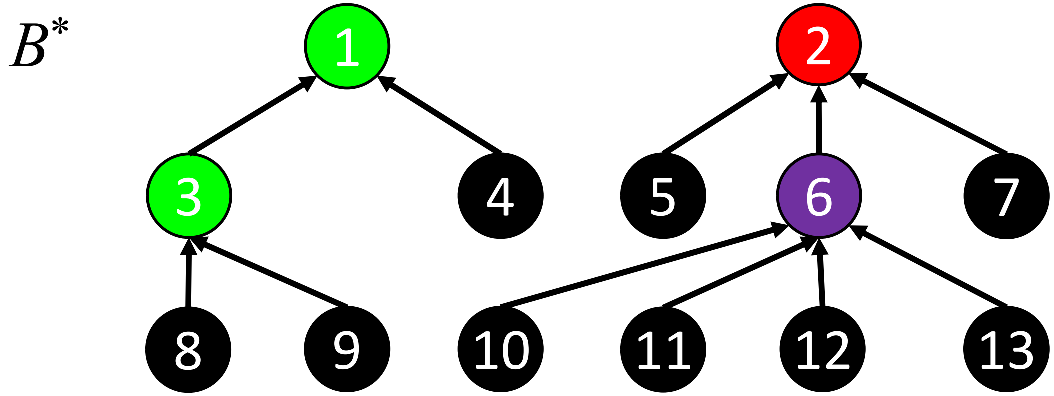
$$\frac{\min(n - a_1|Y_2|, n - a_2(|Y_2| + |Y_2|))}{n - |Y_2| - |Y_3| - |Y_4|}$$

- Let $x = \frac{|Y_2|}{n}$, $y = \frac{|Y_3|}{n}$ and $z = \frac{|Y_4|}{n}$

- The ratio is rewritten as

$$\frac{\min(1 - a_1 x, 1 - a_2(x + y))}{1 - x - y - z}$$

Proof



- Since the total in-degree of $B^* \leq n-1$, we have

  $$2|Y_2| + 3|Y_3| + 4|Y_4| \leq n-1,$$

  and thus $2x + 3y + 4z \leq (n-1)/n \leq 1$

- Similar to Theorem 5.4,

  we summarize our analysis with a mathematical program

Program 3: maximize $\dfrac{\min(1 - a_1 x, 1 - a_2(x+y))}{1-x-y-z}$

subject to $2x + 3y + 4z \leq 1$ and

$x, y, z \geq 0$

Proof

Program 3: maximize $\dfrac{\min(1 - a_1 x, 1 - a_2(x+y))}{1 - x - y - z}$

subject to $2x + 3y + 4z \le 1$ and

$x, y, z \ge 0$

- Note that both the denominator and the numerator of the objective value are positive
- Since the objective value increases as z increases, there is a maximizer with $2x + 3y + 4z = 1$
- Thus, we may rewrite Program 3 by replacing z with $(1 - 2x - 3y) / 4$

Proof

- $\dfrac{\min(1 - a_1 x, 1 - a_2(x+y))}{1 - x - y - z}$

$$= \frac{\min(1 - a_1 x, 1 - a_2(x+y))}{1 - x - y - (1 - 2x - 3y)/4} \qquad \text{(by } z = (1 - 2x - 3y) / 4\text{)}$$

$$= \frac{\min(4 - 4a_1 x, 4 - 4a_2(x+y))}{3 - 2x - y} \qquad (\times \frac{4}{4})$$

Proof

Program 3: maximize $\dfrac{\min(1 - a_1 x, 1 - a_2(x+y))}{1 - x - y - z}$

subject to $2x + 3y + 4z \leq 1$ and

$x, y, z \geq 0$

is rewritten as

Program 4: maximize $\dfrac{\min(4 - 4a_1 x, 4 - 4a_2(x+y))}{3 - 2x - y}$

subject to $2x + 3y \leq 1$ and

$x, y \geq 0$

Proof

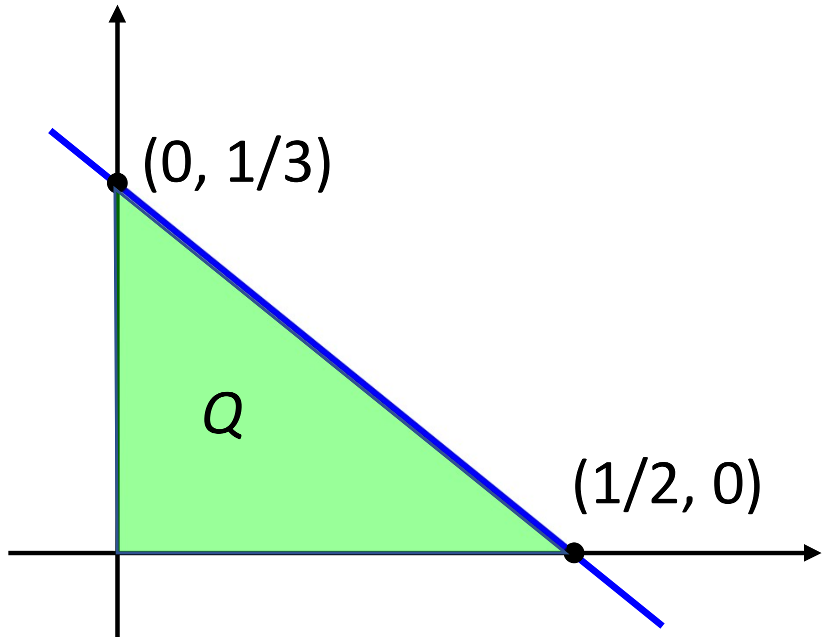
Program 4: maximize $\dfrac{\min(4-4a_1x, 4-4a_2(x+y))}{3-2x-y}$

subject to $2x + 3y \leq 1$ and

$x, y \geq 0$

- Let Q be the feasible region of Program 4

- Note that $x \leq (1/2)$ and $y \leq (1/3)$ in Q



Proof

$$\max \frac{\min(\boxed{4 - 4a_1 x,}\ 4 - 4a_2(x + y))}{3 - 2x - y}$$

$$\text{s.t. } 2x + 3y \le 1,\ x, y \ge 0$$

- Let $f_1(x, y) = \boxed{\dfrac{4 - 4a_1 x}{3 - 2x - y}}$; and

  $f_2(x, y) = \dfrac{4 - 4a_2(x + y)}{3 - 2x - y}$

- Let Q_1 be the subset of feasible points (x, y) with

  $f_1(x, y) < f_2(x, y)$

- Let $Q_2 = Q - Q_1$ be the feasible points with

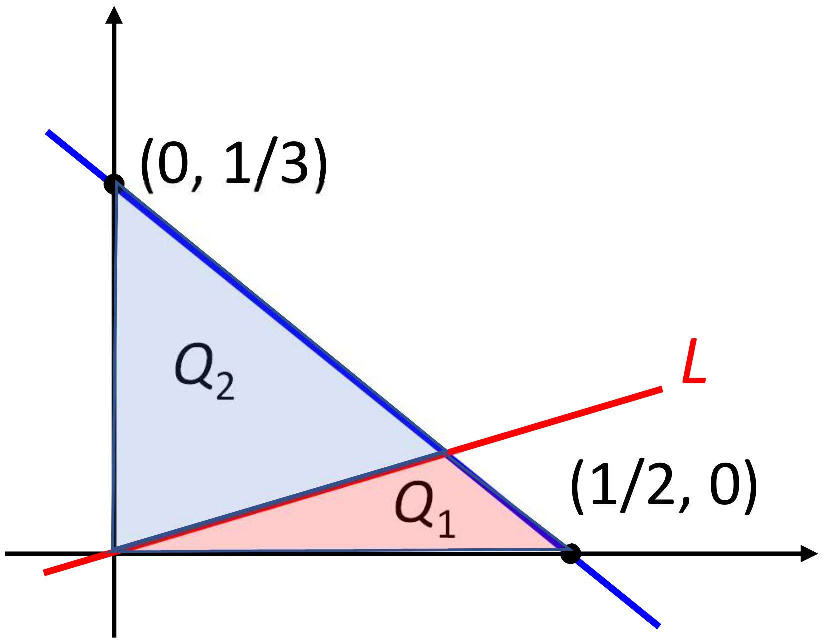
  $f_1(x, y) \ge f_2(x, y)$

Proof

$$\max \frac{\min(\boxed{4-4a_1x}, \ 4-4a_2(x+y))}{3-2x-y}$$

$$\text{s.t. } 2x+3y \le 1,\ x, y \ge 0$$

- The points in Q_1 satisfy
  $$\boxed{4-a_1x} < 4-a_2(x+y),$$
- or equivalently,
  $$(a_1-a_2)x - a_2y > 0$$
- Let L be the line
  $$(a_1-a_2)x - a_2y = 0$$
- Q_1 is the sub-region of Q which is to the right of L



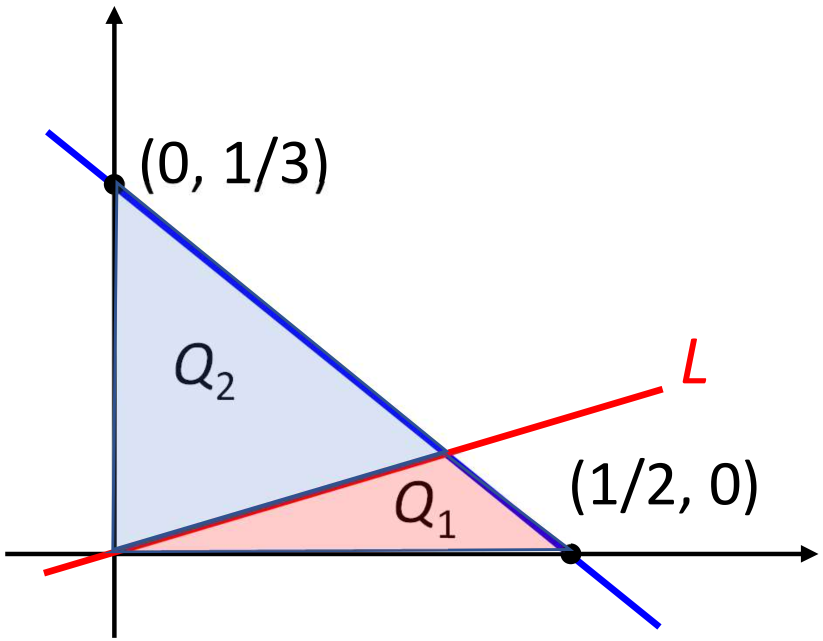
Proof

$$\max \frac{\min(\boxed{4-4a_1x,}\ 4-4a_2(x+y))}{3-2x-y}$$

$$\text{s.t. } 2x+3y \le 1,\ x, y \ge 0$$

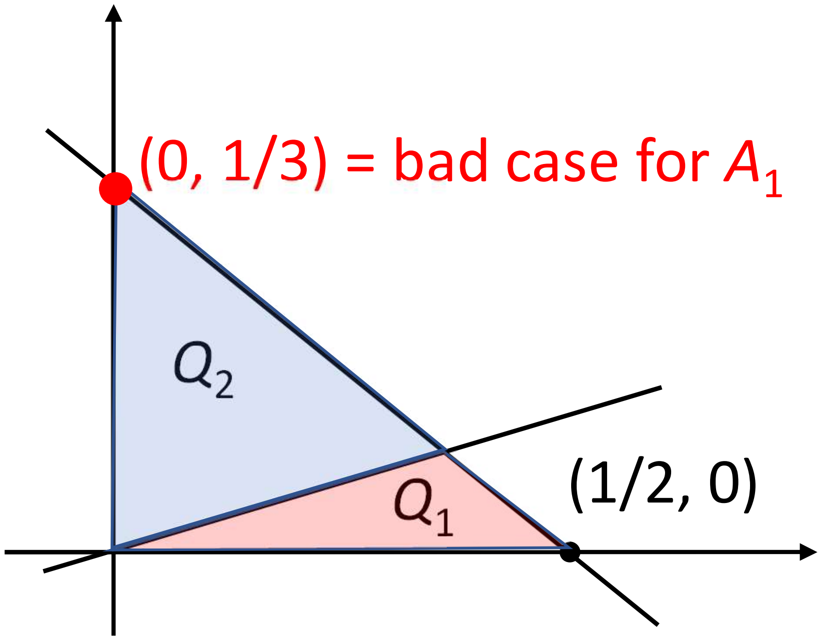
$(a_1 - a_2)x - a_2 y = k$?

variable is k



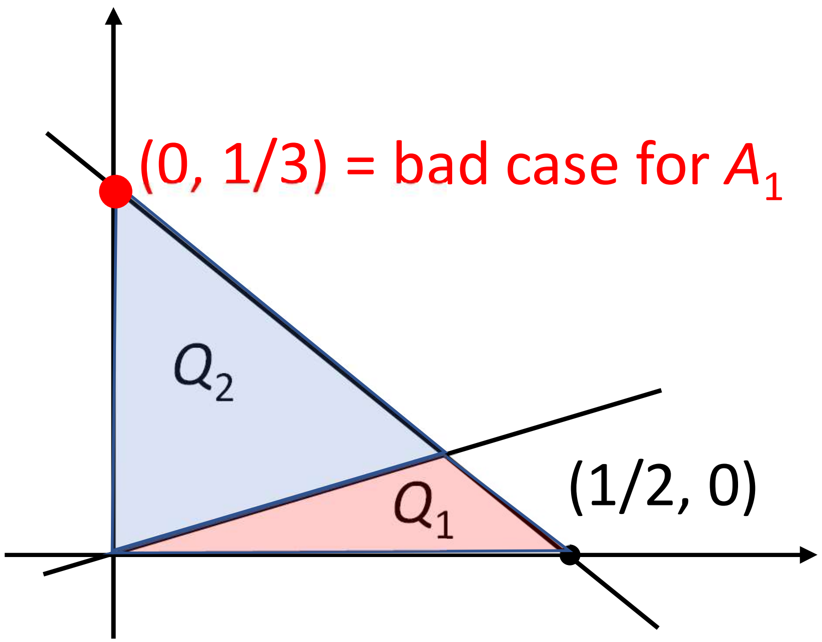
Estimation

- Before proceeding the proof,
  let us give a rough estimation of
  the performance of A_1 and A_2
- The worst case for A_1 happens at $(x, y) = (0, 1/3)$
- It this case, the cost of B_1 / cost of $B^* \approx 3/2 > 4/3$ (our goal)



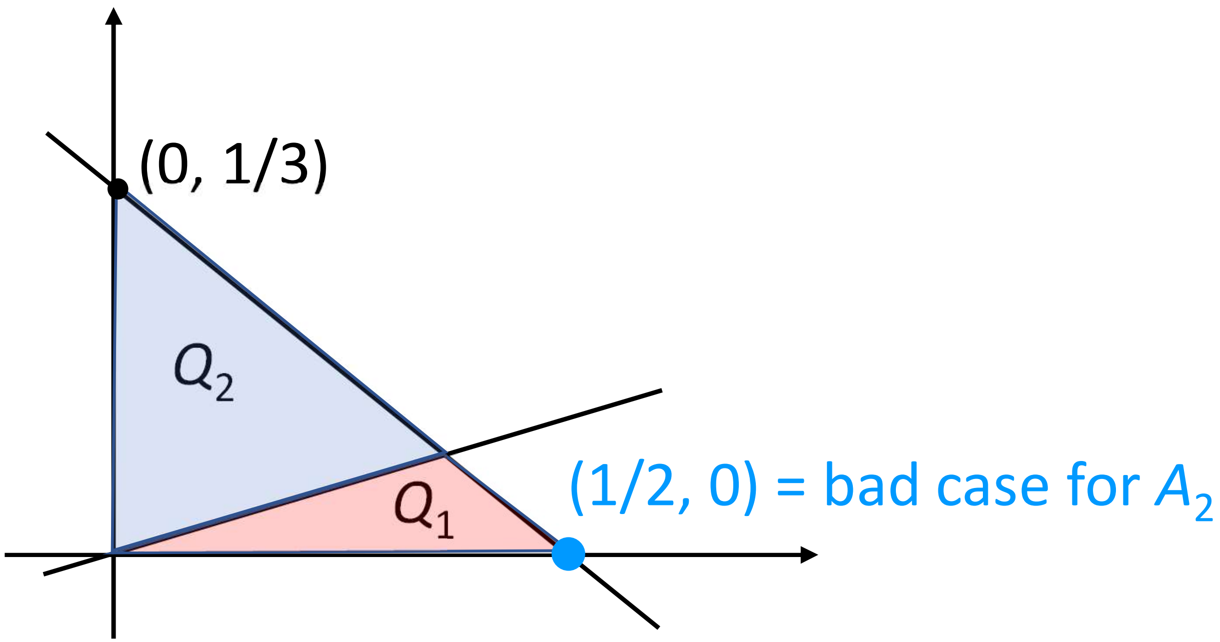
Estimation

- Later, we will show that (0, 1/3) is at Q_2
- That is, in the worst case of A_1,
  A_2 provides a better approximation



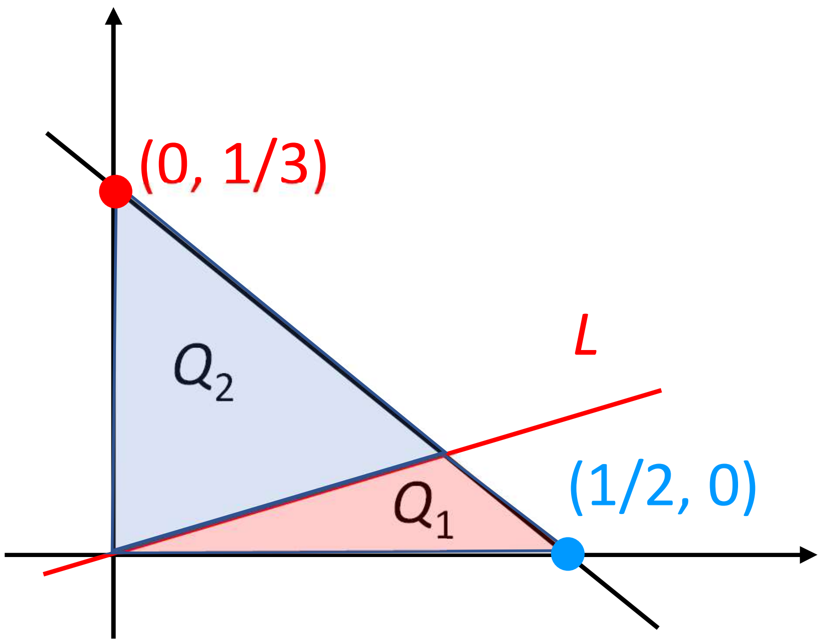
Estimation

- Similarly, the worst case (1/2, 0) for A_2 is in Q_1
- It this case, the cost of B_2 / cost of B^* ≈ 1.4 > 4/3 (our goal)



Estimation

- That is, each of A_1 and A_2 has some bad cases in which the ratio > 4/3
- However, they complement each other
- It can be easily verified that on the line L, both A_1 and A_2 have ratios $\approx 4/3$



Estimation

- Before proceeding the proof,
  let us give a rough estimation of
  the performance of A_1 and A_2

- Recall that a feasible point (x, y) represents
  $|Y_2| / n \approx x$ and $|Y_3| / n \approx y$
  (degree 2) (degree 3)

- A_1 gives an accurate ($\approx 4/3$) approximation to $|Y_2|$ (x)

- A_2 gives a rough ($\approx 5/3$) approximation to $|Y_2| + |Y_3|$ ($x + y$)

Estimation

- The worst case for A_1 is as follows
- In B^*, all reducible vertices have degree 3
- In this case, $x \approx 0$ and $y \approx 1/3$
- It can be verified that cost of B_1 / cost of B^* $\approx 3/2 > 4/3$ (our goal)

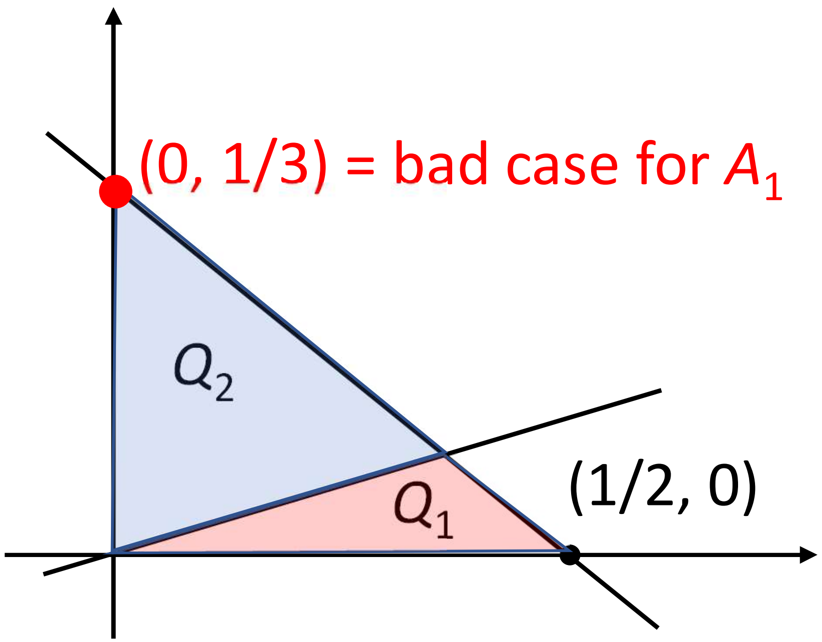
Estimation

$$\max \frac{\min(4 - 4a_1 x,\ 4 - 4a_2(x + y))}{3 - 2x - y}$$

$$\text{s.t. } 2x + 3y \le 1,\ x, y \ge 0$$

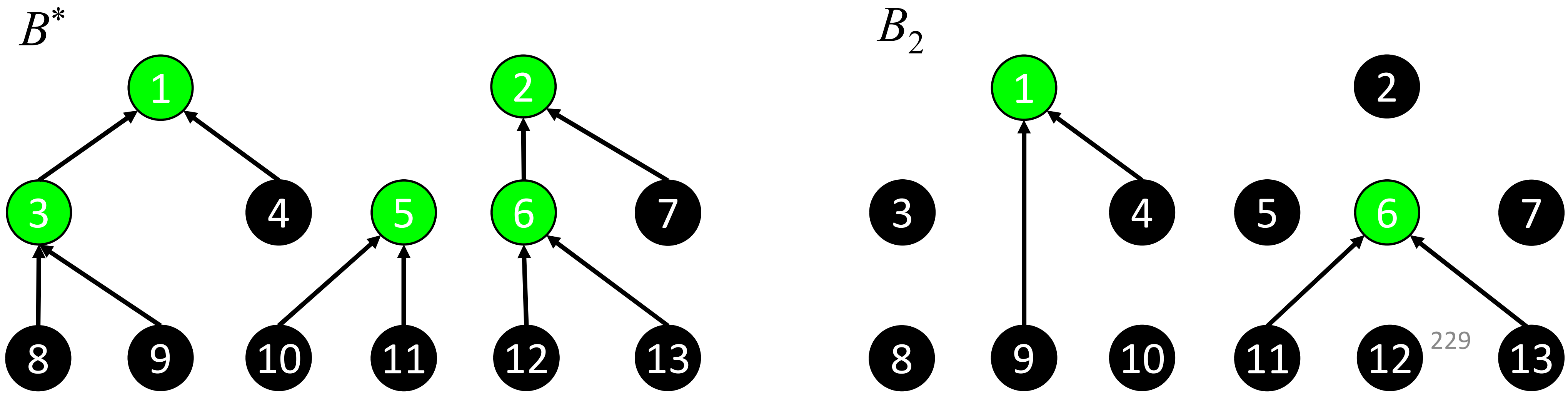
- Later, we will show that the point (0, 1/3) is in Q_2

- That is, in the worst case of A_1,
  A_2 provides a better approximation



Estimation

- The worst case for A_2:
  All reducible vertices have degree 2
- In this case, $x \approx 1/2$ and $y \approx 0$
- It can be verified that cost of B_1 / cost of B^*
  $\approx 7/5 = 1.4 > 4/3$ (our goal)

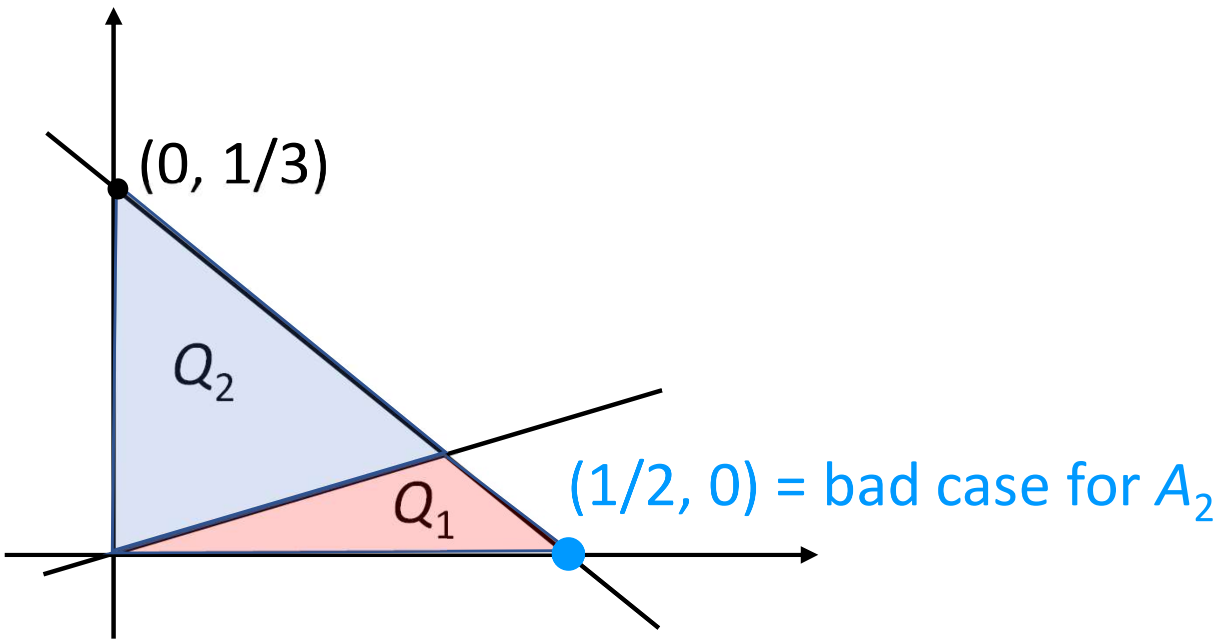


Estimation

- Later, we will show that the point (1/2, 0) is in Q_1

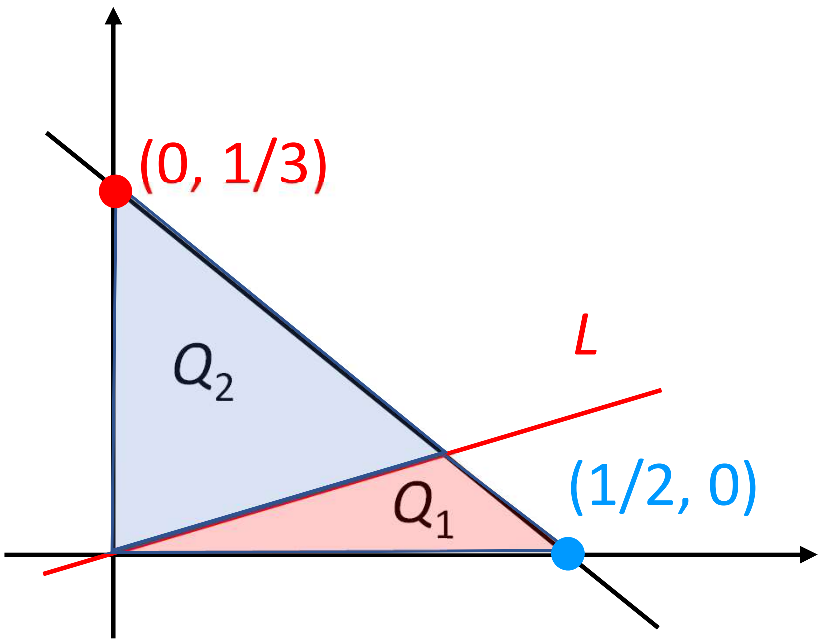
- That is, in the worst case of A_2,

  A_1 provides a better approximation



Estimation

- That is, each of A_1 and A_2 has some bad cases in which the ratio $> 4/3$
- However, they complement each other
- It can be easily verified that on the line L, both A_1 and A_2 have ratios $\approx 4/3$



Proof

- We proceed to determine L
- Recall:

$$\boxed{\begin{gathered} L\colon (a_1 - a_2)x - a_2 y = 0 \\ a_1 = \frac{3}{4} - \frac{1}{2}\delta; \qquad a_2 = \frac{3}{5} - \frac{1}{2}\delta \end{gathered}}$$

- By plugging in the value of a_1 and a_2, L can be rewritten as

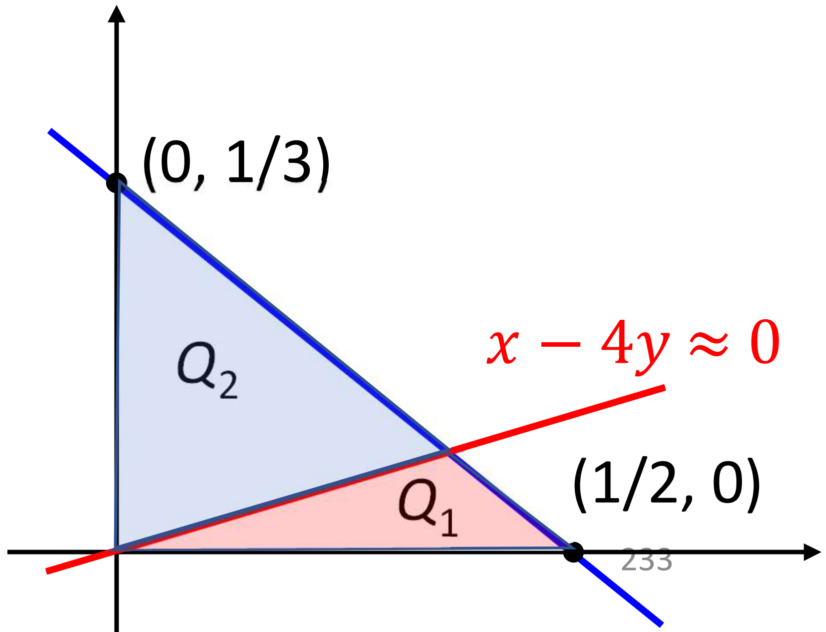
$$\left(\left(\frac{3}{4} - \frac{1}{2}\delta\right) - \left(\frac{3}{5} - \frac{1}{2}\delta\right)\right)x - \left(\frac{3}{5} - \frac{1}{2}\delta\right)y = 0$$

$$\rightarrow \qquad \frac{3}{20}x - \frac{3}{5}y = -\frac{1}{2}\delta y$$

$$\rightarrow \qquad x - 4y = -\frac{20}{6}\delta y = -\frac{10}{3}\delta y \qquad \left(\times \frac{20}{3}\right)$$

Proof

- Remark: When δ is small, L is close to the line $x - 4y = 0$
- Since Q_1 is to the right of L,
  each point in Q_1 satisfies $x - 4y > -\frac{10}{3}\delta y$

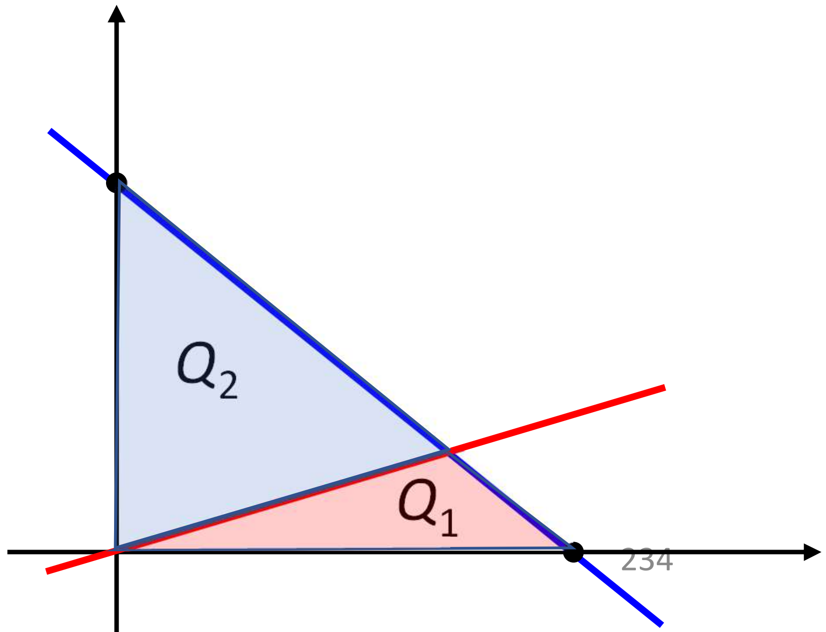


Proof

- Let (x^*, y^*) be the maximizer of Program 4

- Suppose, for the sake of contradiction, that
  the objective value at (x^*, y^*) is greater than $4/3 + \delta$

- Two cases are considered:
  Case 1. $(x^*, y^*) \in Q_1$
  Case 2. $(x^*, y^*) \in Q_2$



Case 1

- Consider Case 1

- Since (x^*, y^*) is to the right of L, we have

$$x^* - 4y^* > -\frac{10}{3}\delta y^*$$

$$\rightarrow x^* - 4y^* > -b, \text{ where } b = \frac{10}{3}\delta y^* \qquad \text{(I1)}$$

- In addition, we know that

  the objective value at (x^*, y^*) is $f_1(x^*, y^*)$

Case 1

$$\max \frac{\min(4 - 4a_1 x,\ 4 - 4a_2(x + y))}{3 - 2x - y}$$

$$\text{s.t. } 2x + 3y \le 1,\ x, y \ge 0$$

- Since $f_1(x^*, y^*) > 4/3 + \delta$,

$$\rightarrow \frac{4 - 4a_1 x^*}{\underbrace{\boxed{3 - 2x^* - y^*}}_{\text{positive}}} > \frac{4+3\delta}{3}$$

$$\times\, 3(3-2x^*-y^*) \rightarrow \cancel{12} - 12a_1 x^* > (\cancel{12} - 8x^* - 4y^*) + 3\delta(3 - 2x^* - y^*)$$

$$\rightarrow -12\underbrace{\left(\frac{3}{4} - \frac{1}{2}\delta\right)}_{= a_1} x^* > -8x^* - 4y^* + 3\delta(3 - 2x^* - y^*)$$

$$\rightarrow -9x^* + \boxed{6\delta x^*} > -8x^* - 4y^* + \boxed{3\delta(3 - 2x^* - y^*)}$$

- Let $s = \boxed{6\delta x^*}$ and $t = \boxed{3\delta(3 - 2x^* - y^*)}$

$$\rightarrow -9x^* + s > -8x^* - 4y^* + t$$

$$\rightarrow s - t > x^* - 4y^* \qquad \text{(I2)}$$

Case 1

$$\boxed{\begin{array}{l}\text{(I1) } x^* - 4y^* > -b \\ \text{(I2) } x^* - 4y^* < s - t\end{array}}$$

- By (I1) and (I2), we have

$$-b < x^* - 4y^* < s - t$$

- Thus, $-b < s - t$ and thus $s - t + b$ is positive

- We now show that $s - t + b$ cannot be positive

(and thus we have a contradiction)

Case 1

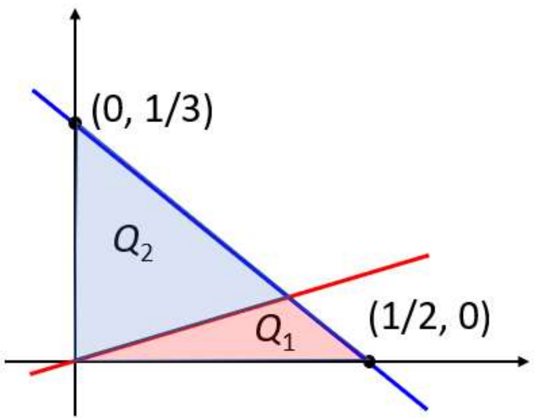
$$s = 6\delta x^*$$
$$t = 3\delta(3 - 2x^* - y^*)$$
$$b = (10/3)\delta y^*$$

- $s - t + b$

$$= (6\delta x^*) - 3\delta(3 - 2x^* - y^*) + \left(\frac{10}{3}\delta y^*\right)$$
$$= \delta(12x^* + \frac{19}{3}y^* - 9)$$

- Since (x^*, y^*) is feasible, $x^* \leq 1/2$ and $y^* \leq 1/3$

$$\rightarrow s - t + b = \delta(12x^* + \frac{19}{3}y^* - 9)$$
$$\leq \delta\left(6 + \frac{19}{9} - 9\right)$$
$$= \frac{-8}{9}\delta \leq 0$$



(contradiction, $s - t + b$ is not positive)

Case 2

- Consider Case 2, in which $p^* \in Q_2$
- The argument is similar
- Recall that Q_2 is to the left of L
- Thus, we have $x^* - 4y^* \leq -b$ (I3)

(where b = (10/3)δy^*)

Case 2

$$\max \frac{\min(4 - 4a_1 x,\ 4 - 4a_2(x + y))}{3 - 2x - y}$$

$$\text{s.t. } 2x + 3y \le 1,\ x, y \ge 0$$

- By our supposition, $f_2(x^*, y^*) > 4/3 + \delta$

$$\to \frac{4 - 4a_2(x^* + y^*)}{3 - 2x^* - y^*} > \frac{4 + 3\delta}{3} \qquad \times\, 3(3 - 2x^* - y^*)$$

$$\to \cancel{12} - 12a_2(x^* + y^*) > (\cancel{12} - 8x^* - 4y^*) \boxed{+\, t}$$

$$\to -12\underbrace{\left(\frac{3}{5} - \frac{1}{2}\delta\right)}_{= a_2}(x^* + y^*) > -8x^* - 4y^* + t$$

$$\to \left(6\delta - \frac{36}{5}\right)(x^* + y^*) > -8x^* - 4y^* + t$$

$$\to 6\delta(x^* + y^*) + \left(8 - \frac{36}{5}\right)x^* + \left(4 - \frac{36}{5}\right)y^* > t$$

$$\to 6\delta(x^* + y^*) + \frac{4}{5}x^* - \frac{16}{5}y^* > t$$

$$\boxed{\text{Recall: } t = 3\delta(3 - 2x^* - y^*)}$$

Case 2

- In last page: $6\delta(x^* + y^*) + \frac{4}{5}x^* - \frac{16}{5}y^* > t$

- Multiply both side by $\frac{5}{4}$, we have

$$\boxed{\frac{15}{2}\delta(x^* + y^*)} + x^* - 4y^* > \frac{5}{4}t$$

- Let $c = \boxed{\frac{15}{2}\delta(x^* + y^*)}$

$\rightarrow\ x^* - 4y^* > -c + (5/4)t$ (I4)

Case 2

- Recall:

  (I3) $x^* - 4y^* \leq -b$

  (I4) $x^* - 4y^* > -c + (5/4)t$

- Thus, $-b > -c + (5/4)t$

  $\rightarrow c - b - (5/4)t$ is positive

- Again, we will show that $a - b - (5/4)t$ is non-positive

$$a = \frac{15}{2}\delta(x^* + y^*)$$

$$b = \frac{10}{3}\delta y^*$$

$$t = 3\delta(3 - 2x^* - y^*)$$

Case 2

- $c \overbrace{\boxed{-\,b}}^{\leq 0} - (5/4)t$

$$\leq c - (5/4)t$$

$$= \left[\frac{15}{2}\delta(x^* + y^*)\right] - \frac{5}{4}\left[3\delta(3 - 2x^* - y^*)\right]$$

$$= \delta \times \left(\left(\frac{15}{2} + \frac{15}{2}\right) \times x^* + \left(\frac{15}{2} + \frac{15}{4}\right) \times y^*\right.$$

$$\left. + \left(-\frac{45}{4}\right) \times 1\right)$$

$$= \delta\left(15x^* + \frac{45}{4}y^* - \frac{45}{4}\right)$$

Case 2

- Last page: $a - b - (5/4)t \le \delta(15x^* + \frac{45}{4}y^* - \frac{45}{4})$
- Since (x^*, y^*) is feasible, $x^* \le 1/2$ and $y^* \le 1/3$
- Thus, $\delta(15x^* + \frac{45}{4}y^* - \frac{45}{4})$

$$\le \delta\left(\frac{15}{2} + \frac{15}{4} - \frac{45}{4}\right)$$

$$= \delta\left(\frac{30}{4} + \frac{15}{4} - \frac{45}{4}\right) = 0$$

(a contradiction)

- This implies that $a - b - (5/4)t$ is not positive

Proof

- The obtained contradiction shows that
  (the maximum of Program 4) $\leq 4/3 + \delta$

- Thus, Algorithm 2 guarantees
  an approximation ratio of $4/3 + \delta$

- This completes the proof

Remark 1

- A natural extension of Algorithm 2 is to
  consider not only $\mathcal{F}_2$, $\mathcal{F}_3$,
  but also $\mathcal{F}_4$, $\mathcal{F}_5$, ..., and so on

- However, it seems to the author that
  the approximation ratio of this algorithm is also $4/3 + \delta$

- The reason is as follows

Remark 1 (cont'd)

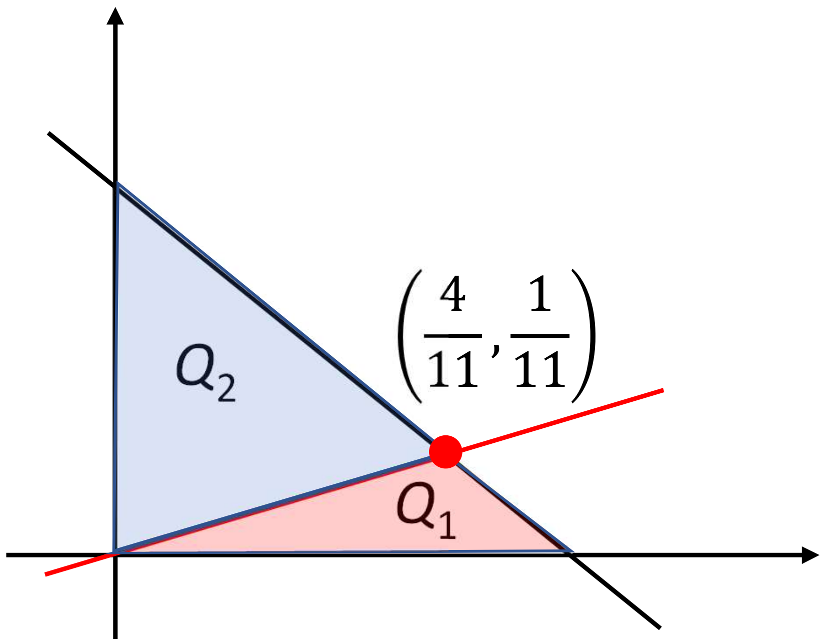
degree 2

- When $|Y_2| \approx \frac{4}{11}n$, $|Y_3| \approx \frac{1}{11}n$, and $|Y_4| \approx 0$,

  both B_1 and B_2 cost about $\frac{4}{3}$ times the optimal cost

- Since $|Y_4| = 0$, in our analysis,

  $\mathcal{F}_4$ provides no additional information



Remark 2

- Although Algorithm 2 is extremely inefficient,
  our technique can be used to
  design practical approx. algorithms for MDCRSP

- For example, we may replace A_1 and A_2 by
  more efficient approximation algorithms
  for 3-SP and 4-SP, respectively

- The interested reader is referred to [?]
  for a survey of approximation algorithms for k-SP

Future work

- Consider the following problem

- Name: d-MDCRSP
- Statement: Given a matrix M, find the minimum cost branching whose maximum in-degree is d

- Algorithm 1 can be seen as an approximation algorithm for 2-MDCRSP

Future work

- If d-MDCRSP is polynomial time solvable for d = 2 and 3, the ratio of Algorithm 2 can be improved to (5/4)
- In addition, there will be a simple (4/3)-approximation algorithm
- Thus, we leave as an open problem to find a polynomial-time algorithm 2/3-MDCRSP
- Recall that for d >= ???, the problem is NP-hard

Future work

- Other future works:

- Faster FPT-time algorithm for MSRP
- Better kernel size for MSRP
- Constant approximation for MSRP
- Improve the approx. ratio for MDCRSP
- Improve the time complexity of our MDCRSP algorithm
- Approximation lower bound for MDCRSP

Case 2

$$a = \frac{15}{2}\delta(x^* + y^*)$$

$$b = \frac{10}{3}\delta y^*$$

$$t = 3\delta(3 - 2x^* - y^*)$$

- $a - b - (5/4)t$

$$= [\frac{15}{2}\delta(x^* + y^*)] - [\frac{10}{3}\delta y^*] - \frac{5}{4}[3\delta(3 - 2x^* - y^*)]$$

$$= \delta \times ((\quad \frac{15}{2} + \frac{15}{2} \quad) \times x^* + (\quad \frac{15}{2} - \frac{10}{3} + \frac{15}{4} \quad) \times y^*$$

$$+ \quad (\quad -\frac{45}{4} \quad))$$

$$= \delta(15x^* + \frac{95}{12}y^* - \frac{45}{4})$$

Remark

- The two cases:
  Case 1. $(x^*, y^*) \in Q_1$
  Case 2. $(x^*, y^*) \in Q_2$

- To obtain a contradiction:

  - In Case 1, we will show that (x^*, y^*) must be in Q_2

  - In Case 2, we will show that (x^*, y^*) must be in Q_1

Remark

- The proof of Theorem ?? can be extended to analyze the algorithms of the following kind:

- Algorithm 1:
- Input: a matrix M, an integer $d \geq 2$
  - Step 1. compute D_M and E
  - Step 2. compute $\mathcal{F}_d$
  - Step 3. use an approximation algorithm for $(d + 1)$-SP to find a packing $\mathcal{P}$ of $(E, \mathcal{F}_d)$
  - Step 4. transform $\mathcal{P}$ to a branching B
  - Step 5. output B

Theorem

- *Proof*. Let $\rho_1 = 4/(3 + 12\delta)$ be a positive number < 4 / 3

- By Theorem ??, there exists
  a ρ_1-approximation algorithm A_1 for 3-SP

- Let $a_1 = 1 / \rho_1$a ((4 + 1) / 3 + ε)-approximation algorithm A_2 for 4-SP

- Let $\rho_1 = 4/3 + \varepsilon$ and $\rho_2 = 5/3 + \varepsilon$ be, respectively,
  the approx. ratios for A_1 and A_2

Theorem

- *Proof*. Let $\rho_1 = 4/(3 + 12\delta)$ be a positive number $< 4 / 3$

- By Theorem ??, there exist:
  - a $((3 + 1) / 3 + \varepsilon)$-approximation algorithm A_1 for 3-SP
  - a $((4 + 1) / 3 + \varepsilon)$-approximation algorithm A_2 for 4-SP

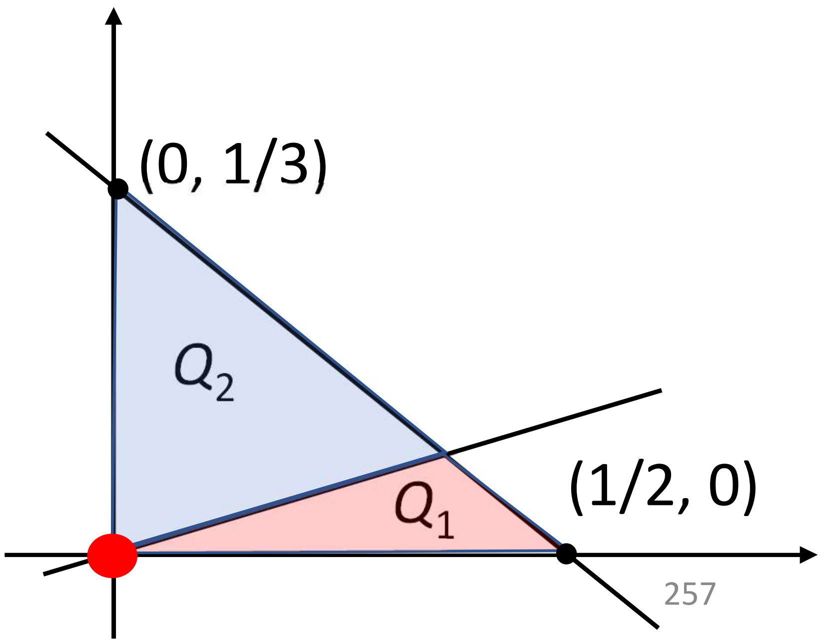
- Let $\rho_1 = 4/3 + \varepsilon$ and $\rho_2 = 5/3 + \varepsilon$ be, respectively, the approx. ratios for A_1 and A_2

Estimation

$$\max \frac{\min(4 - 4a_1 x,\ 4 - 4a_2(x + y))}{3 - 2x - y}$$

$$\text{s.t. } 2x + 3y \le 1,\ x, y \ge 0$$

- For case where $|Y_2| = |Y_3| = 0$, $|Y_4| \approx n/4$:
- In this case, $x \approx y \approx 0$
- Both algorithm guarantees a ratio of 4/3 (our goal)
- Surprisingly, this is not the worst case for either algorithm



Estimation

$$\max \frac{\min(4 - 4a_1 x,\ 4 - 4a_2(x + y))}{3 - 2x - y}$$

$$\text{s.t. } 2x + 3y \le 1,\ x, y \ge 0$$

- Recall that L is close to the line $x - 4y = 0$
- The performance of A_1 and A_2 coincides on L
- That is, when $y = 4x$:
  - B_1 guarantees $(4 - 4a_1x) / (3 - 2x - y)$
    $\approx (4 - 12y) / (3 - 9y) = 4/3$,
  - and so does B_2
  - Each of B_1 and B_2 gurantees a ratio of 4/3

Proof

- For each pair of vertices $u, v \in V(D_M)$:
  - Let $p = u \cup v$
  - If $p \in V(D_M)$, $p \neq u$ and $p \neq v$,
    add $\{p^{(\text{in})}, u^{(\text{out})}, v^{(\text{out})}\}$ to $\mathcal{F}_2$
- We need an efficient data structure to check if $p \in V(D_M)$