Inferring a Graph from Path Frequency

Tatsuya Akutsu[1,2] & Daiji Fukagawa[2]

[1] Institute for Chemical Research, Kyoto Univ., Japan

[2] Graduate School of Informatics, Kyoto Univ., Japan

Outline

- Introduction
  - String inference from spectrum feature (SISF)
  - Graph inference from path frequency (GIPF)
  - Optimization versions (SISF-M, GIPF-M)

- Algorithms for special cases
- Complexity results

  Strong NP-completeness of GIPF in general case

  (Reduction from 3-PARTITION)
- Conclusion

Motivation

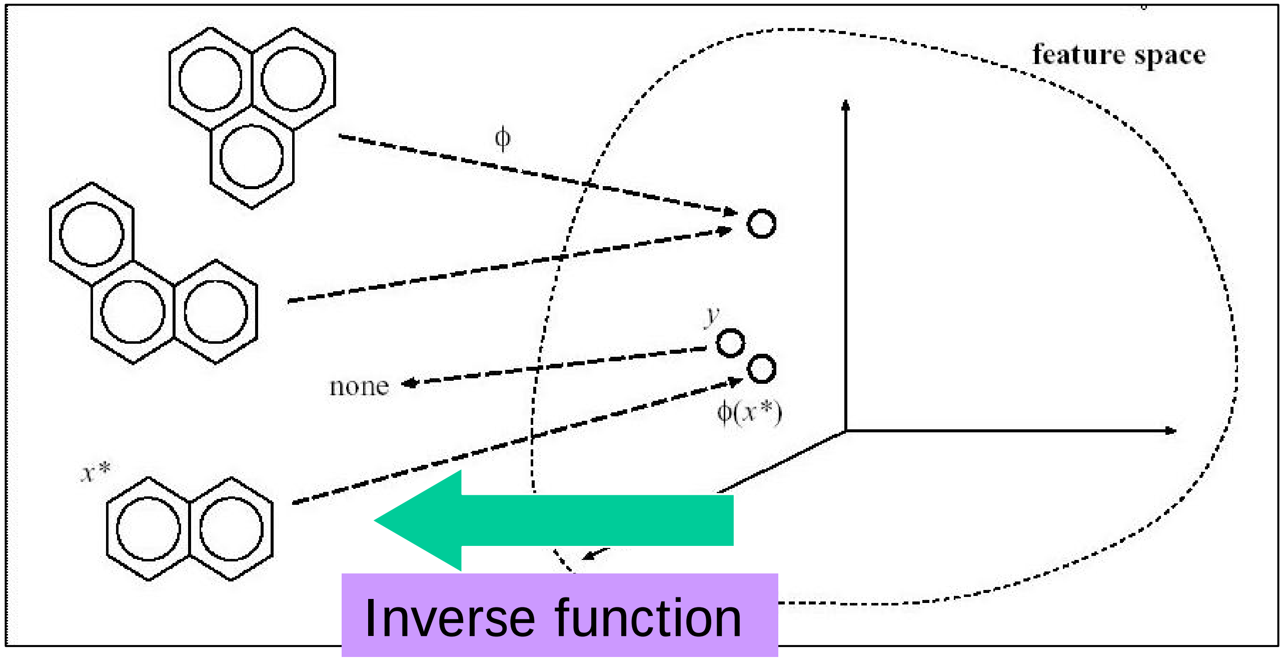
- **<u>Kernel methods (e.g. Support Vector Machine) have been applied to various problems. In kernel methods,</u>**

  Data (sequences, chemical compounds,…) ➔ Feature vector

- **<u>This work: we consider reverse direction, i.e.,</u>**

  Feature vector ➔ Data

  May be useful for designing new sequences/chemical compounds

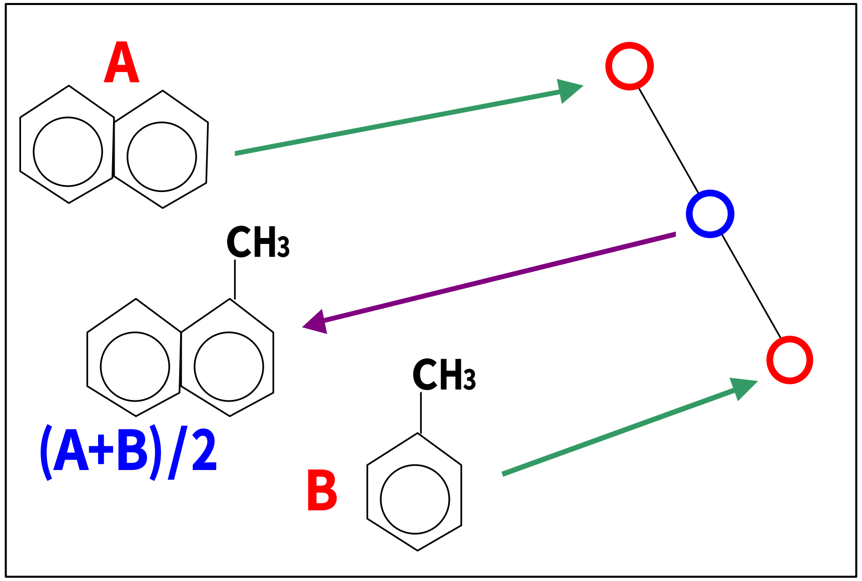


Motivation (continued)

Potential application:

drug design

- For example, design a new compound which is the middle of known compounds A and B



Related work

- Kernel PCA + regression [Bakir,Weston,Scölkopf 2004]
- Graph pre-image [Bakir,Zien,Tsuda 2004]
- But, no complexity studies

graph inference problem

Graph inference from path frequency

Given path frequency vector *v*, infer the original graph whose feature vector (=path frequency) is equal to *v* (or closest to *v*).

(length) (path: occurrence)

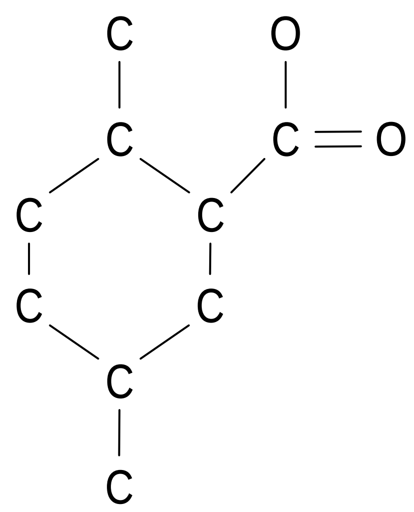
0 --- C x 9, O x 2

1 --- CC x 18, CO x 2, OC x 2

2 --- CCC x 24, CCO x 2, OCC x 2, OCO x 2

3 --- CCCC x 26, CCCO x 4, OCCC x 4

: : : : :



Spectrum Feature for Strings [Leslie *et al.* 02]

For a string S,

- Spectrum feature of level k is a frequency vector of all possible k-grams.

e.g. spectrum feature of level 2 for 'aababb':

aa	x 1
ab	x 2
ba	x 1
bb	x 1

➔ f_2('aababb')=(1,2,1,1)

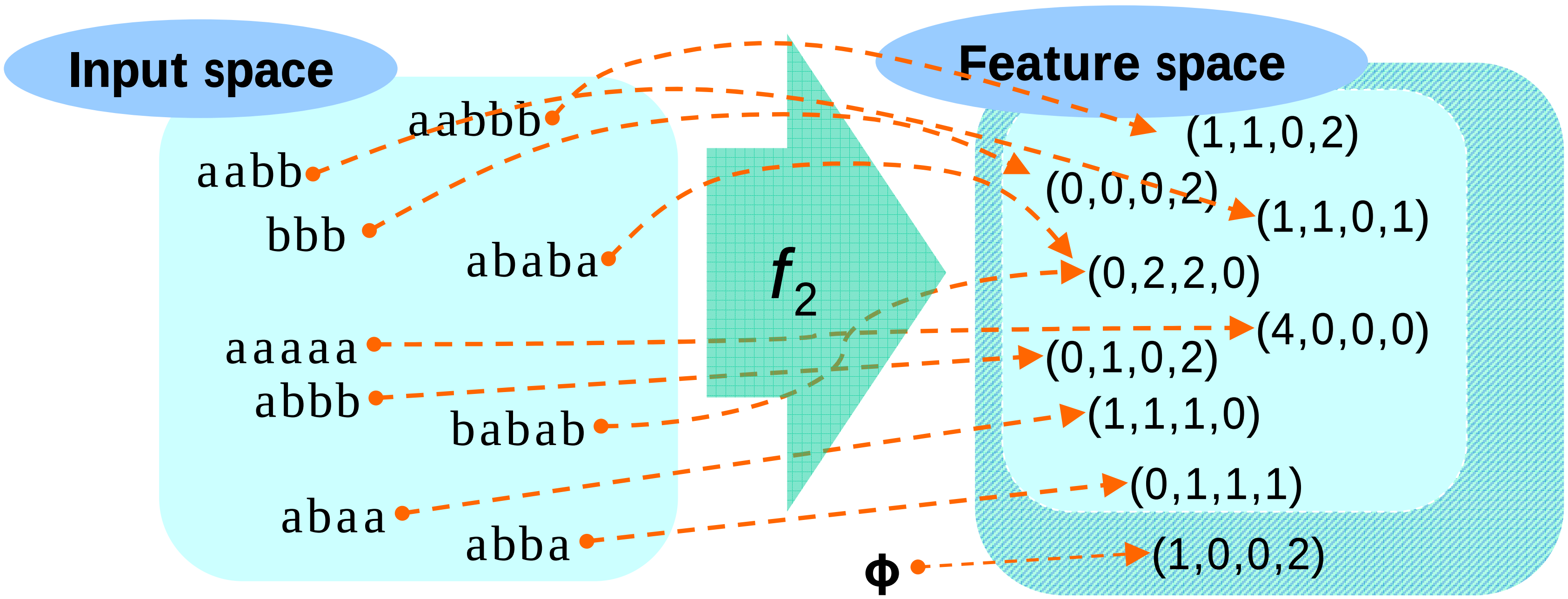
Spectrum Feature for Strings [Leslie *et al.* 02]

f_K: mapping from input space to feature space

$$f_K(s) = \left(occ(t,s)\right)_{t \in \Sigma^K}$$

K: level (>0)
Σ : alphabet

where *occ*(*t*,*s*) is # of occurrences of a substring *t* in a string *s*.



Problem 1

SISF: *String Inference from Spectrum Feature*

Input: an integer K, feature vector $v = (v_t)_{t \in \Sigma^K}$

Output: a string s which, if it exists, satisfies $f_K(s) = v$, otherwise "no solution."

Ex) Σ={a,b}, K=2, $v=(v_{aa},v_{ab},v_{ba},v_{bb})=(1,1,0,2)$

$|s|=(v_{aa}+v_{ab}+v_{ba}+v_{bb})+(K-1)=(1+1+0+2)+1=5$

$$\left.\begin{array}{l} f_2(\text{'aaaaa'})=(4,0,0,0) \\ f_2(\text{'aaaab'})=(3,1,0,0) \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ f_2(\text{'aabbb'})=\boxed{(1,1,0,2)} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ f_2(\text{'bbbbb'})=(0,0,0,4) \end{array}\right\}$$

solution: s='aabbb'

Solutions may not be unique:

v=(1,1,1,1) → 4 solutions

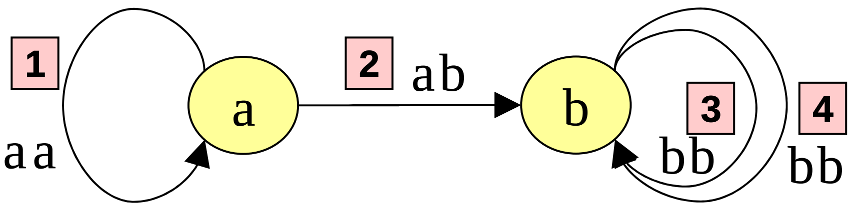
v=(1,2,2,1) → 12 solutions

Linear time algorithm for SISF

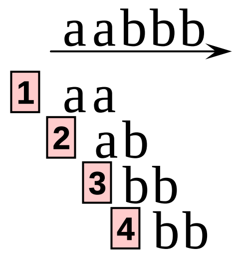
Reduction to Eulerian graph problem [Pevzner]

$f_K(s)=v$ for some s ⇔ G_v has a Eulerian path

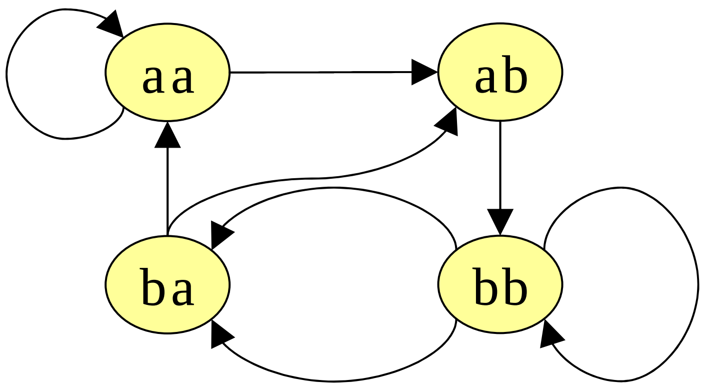
Ex.1) K=2, $v=(v_{aa},v_{ab},v_{ba},v_{bb})=(1,1,0,2)$



solution



Ex.2) K=3, $v=(v_{aaa},v_{aab},v_{aba},v_{abb},v_{baa},v_{bab},v_{bba},v_{bbb})=(1,1,0,1,1,1,2,1)$



2 solutions

$\begin{cases} \text{bbaaabbbab} \\ \text{bbbaaabbab} \end{cases}$

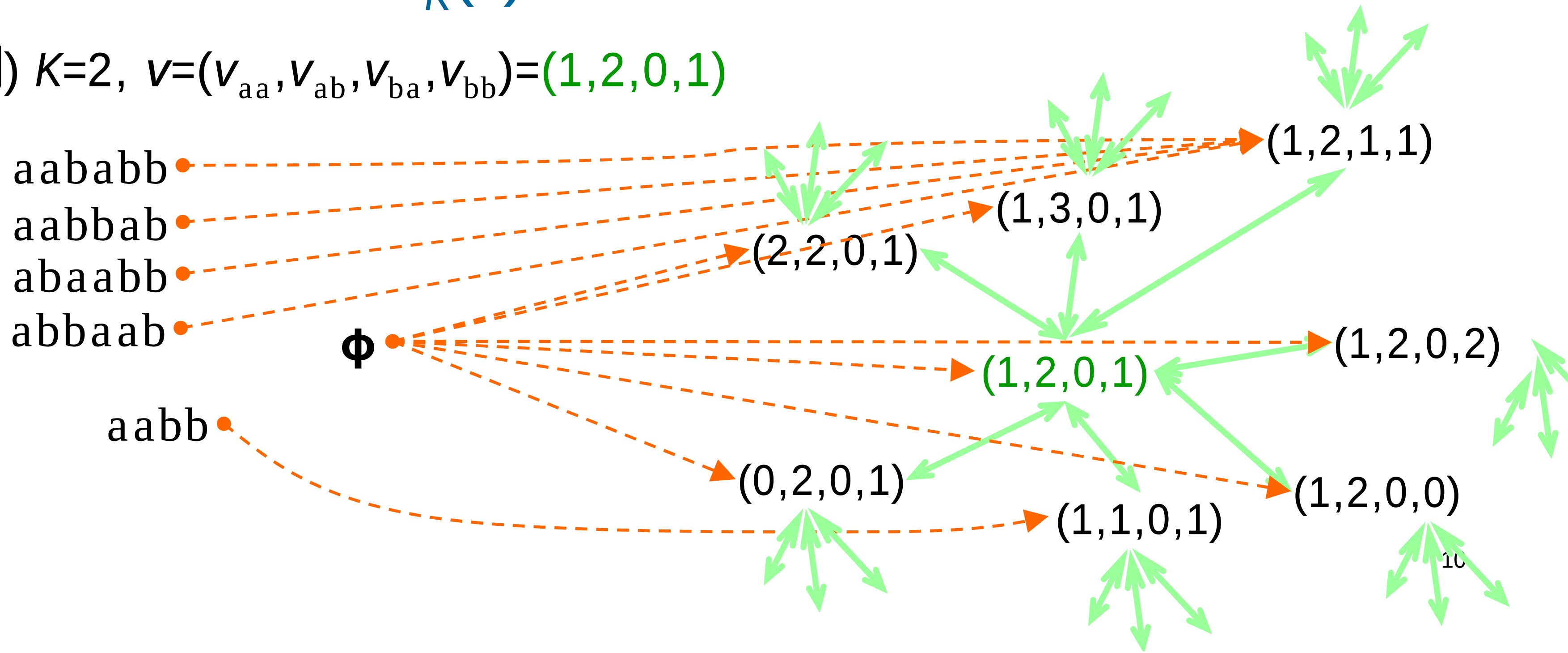
Problem 2

SISF-M: *SISF with the Minimum Error*

Input: an integer K, feature vector $v = (v_t)_{t \in \Sigma^K}$

Output: a string s which minimizes the distance between $f_K(s)$ and v

例) K=2, $v=(v_{aa},v_{ab},v_{ba},v_{bb})$=(1,2,0,1)



Algorithm for SISF-M

- It seems difficult to apply Eulerian path technique.
- Thus, we employ another approach based on Dynamic programming.
- The algorithm is a special case of the graph inference algorithm.

Feature vector for graphs: Path Frequency (c.f. Marginalized Graph Kernel)

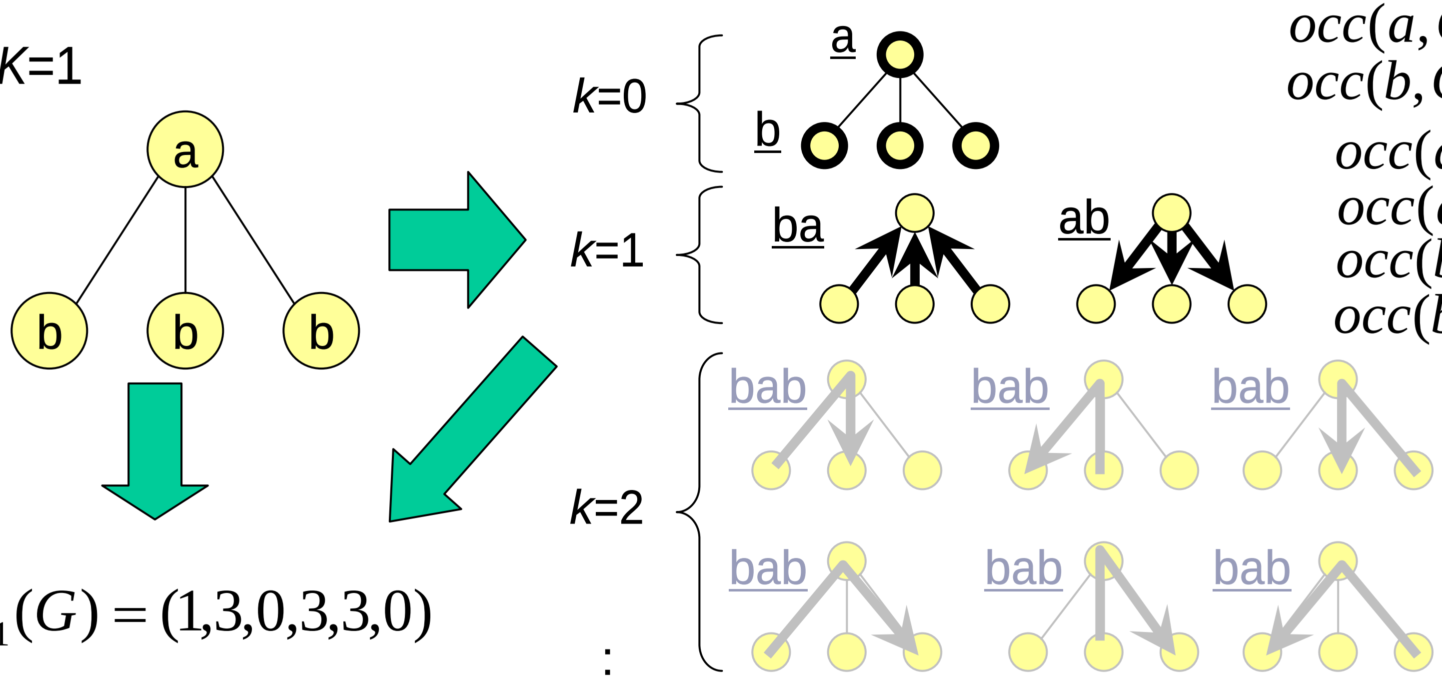
Feature vector f_K: G $\rightarrow \mathbf{N}^{\Sigma^{\leqq K}}$

$$f_K(G) = \left(occ(t,G)\right)_{t \in \Sigma^{\leq K}}$$

K: level(>0)
Σ : alphabet
G: graph

where *occ*(*t*,*G*) is # of occurrences of paths labeled with *t* in a graph *G*

e.g.) *K*=1



$$occ(a,G) = 1$$
$$occ(b,G) = 3$$
$$occ(aa,G) = 0$$
$$occ(ab,G) = 3$$
$$occ(ba,G) = 3$$
$$occ(bb,G) = 0$$

$$f_1(G) = (1,3,0,3,3,0)$$

Problem 3 & 4

GIPF: *Graph Inference from Path Frequency*

Input: an integer K, feature vector $v = (v_t)_{t \in \Sigma^K}$

Output: a graph G which satisfies, if it exists, $f_K(G) = v$ otherwise "no solution."

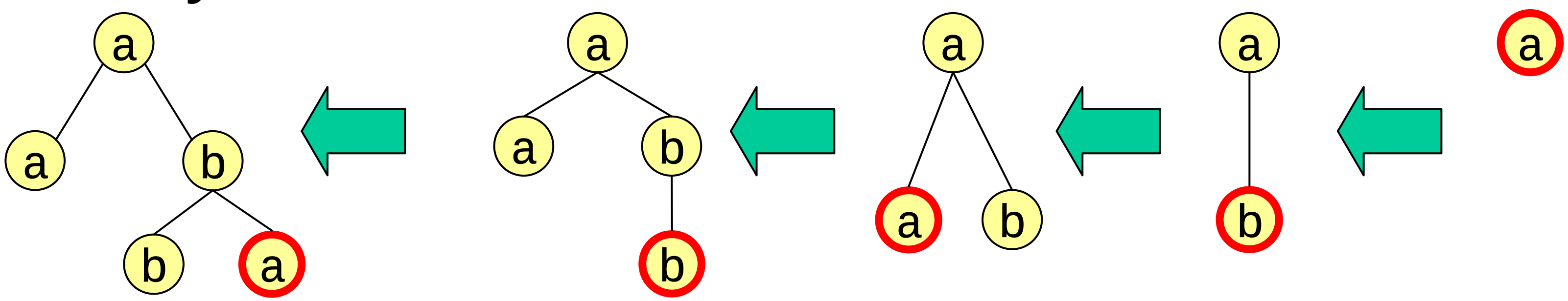
GIPF-M: *GIPF with Minimum Error*

Input: an integer K, feature vector $v = (v_t)_{t \in \Sigma^K}$

Output: a graph G, which minimizes the distance between $f_K(G)$ and v

Dynamic Programming for restricted GIPF (1) Trees, K=1, fixed Σ

Any tree can be constructed by inserting a leaf one by one.

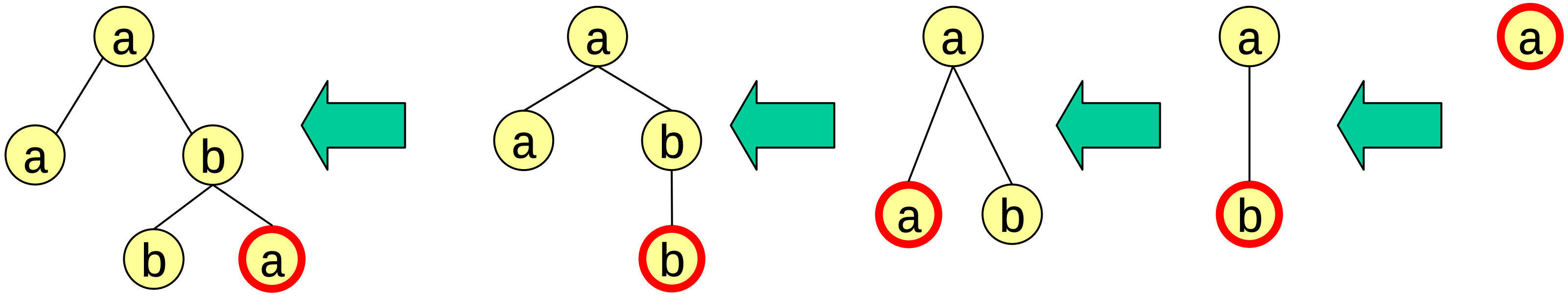


D(v)=1 iff. There exists a tree T s.t. $f_v(T)=v$

$$
\begin{aligned}
&D(n_{\mathrm{a}}, n_{\mathrm{b}}, n_{\mathrm{aa}}, n_{\mathrm{ab}}, n_{\mathrm{ba}}, n_{\mathrm{bb}}) = 1 \Leftrightarrow \\
&\quad \left(D(n_{\mathrm{a}} - 1, n_{\mathrm{b}}, n_{\mathrm{aa}} - 2, n_{\mathrm{ab}}, n_{\mathrm{ba}}, n_{\mathrm{bb}}) = 1\right) \vee \\
&\quad \left(D(n_{\mathrm{a}} - 1, n_{\mathrm{b}}, n_{\mathrm{aa}}, n_{\mathrm{ab}} - 1, n_{\mathrm{ba}} - 1, n_{\mathrm{bb}}) = 1\right) \vee \\
&\quad \left(D(n_{\mathrm{a}}, n_{\mathrm{b}} - 1, n_{\mathrm{aa}}, n_{\mathrm{ab}} - 1, n_{\mathrm{ba}} - 1, n_{\mathrm{bb}}) = 1\right) \vee \\
&\quad \left(D(n_{\mathrm{a}}, n_{\mathrm{b}} - 1, n_{\mathrm{aa}}, n_{\mathrm{ab}}, n_{\mathrm{ba}}, n_{\mathrm{bb}} - 2) = 1\right)
\end{aligned}
$$

Dynamic Programming for restricted GIPF (1) Trees, K=1, fixed Σ (cont'd.)

Any tree can be constructed by inserting a leaf one by one.



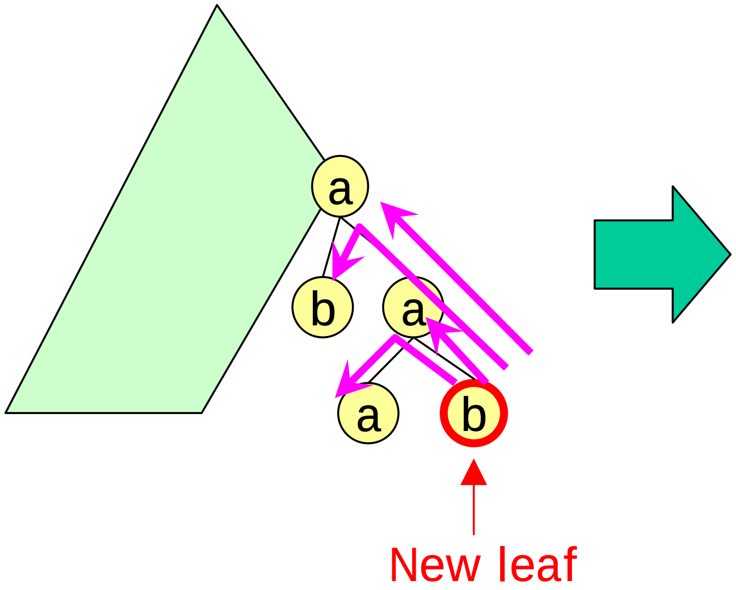
D(v)=1 iff. There exists a tree T s.t. $f_v(T)=v$

《Theorem》 GIPF for trees is solved in polynomial time in n (the size of tree) if K=1 and a fixed alphabet.

→ GIPF-M is also solved by searching in this table.

Dynamic Programming for restricted GIPF (2) Trees, fixed K, Δ, Σ

- Extension of DP for K=1 (not straightforward)
- More complicated data structure than K=1.



When a new leaf is added, much more new paths appear

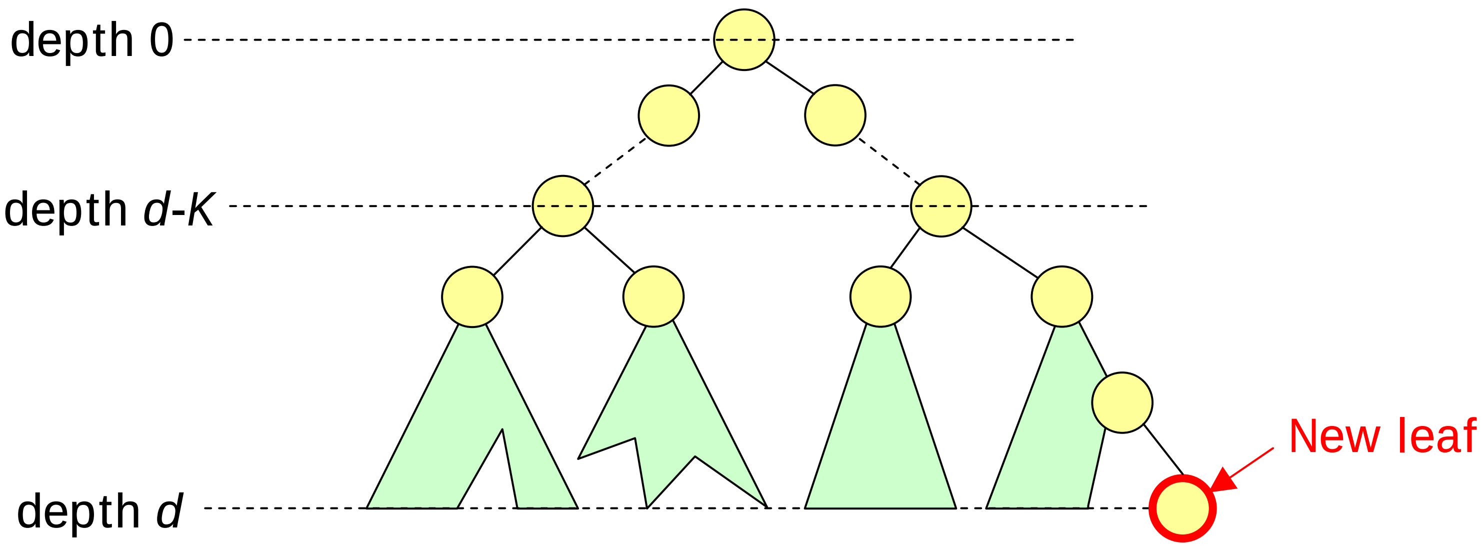
→ O(Δ^K) new paths

Dynamic Programming for restricted GIPF (2) Trees, fixed K, Δ, Σ (cont'd)

- Extended DP table: D(v,e,d)

  v: feature vector, e: paths around leaves, d: depth

- K, Δ, Σ are fixed, so is the size of e.



《Theorem》 GIPF (and GIPF-M) for trees is solved in polynomial time in n if K, Δ, Σ are all fixed.

Strong NP-completeness of GIPF

- **GIPF is strongly NP-complete** even if the underlying graph is a tree and K=3

(We improved the result from that in the proceedings, where this result was shown for non-tree graphs)

- Reduction from 3-PARTITION problem

Reduction from: 3-PARTITION, $P=(X,w,B)$

- $X=\{x_1,\dots,x_{3m}\}$, $w(x_i)=w_i$, $\sum w_i=Bm$

Reduction into: GIPF, $Q=(v,K)$

- $\Sigma=\{a,b,c,d,y\}\cup\{x_1,\dots,x_{3m}\}\cup\{A_1,\dots,A_m\}$
- K=3
- $v(s)$=O(poly(m+B)) for every $s\in\Sigma^K$
- $|\{s\in\Sigma^K \mid v(s)$ is non-zero$\}|$ = O(poly(m+B))

Strong NP-completeness of GIPF

《Theorem》 GIPF is strongly NP-complete, even if K=3 and the underlying graph is a tree.

(Proof)

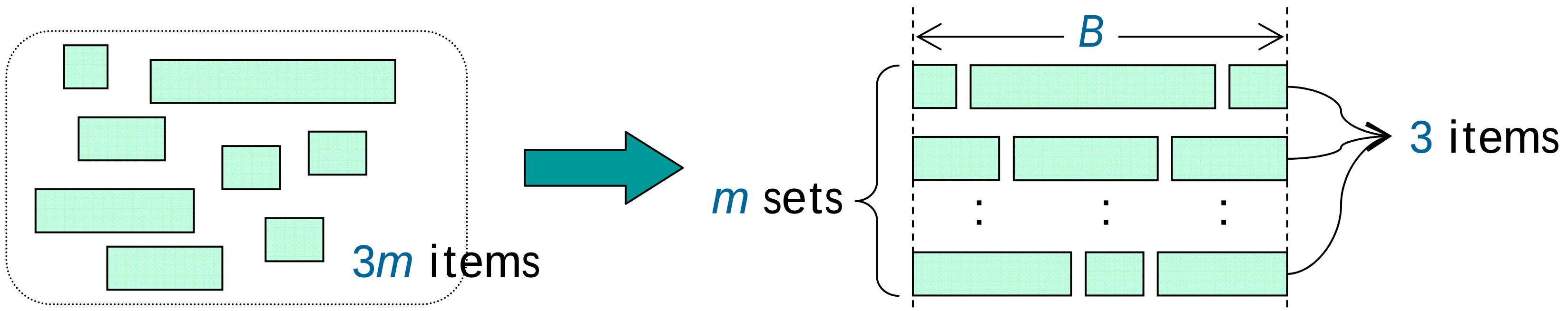
- Reduction from 3-PARTITION can be done in poly(m+B) time and thus its size is bounded by poly(m+B).
- GIPF Q is 'yes' ⇔ 3-PARTITION P is 'yes'

3-PARTITION problem [Garey&Johnson]

3-PARTITION: strongly NP-complete problem

Input: a set $X=\{x_1,\ldots,x_{3m}\}$ of $3m$ items, for each x_i a weight $w(x_i)$, s.t. $\Sigma_i w(x_i)=mB$

Output: 'yes' if there exist a partition $A_1,\ldots,A_m$ of X s.t. $|A_h|=3$ and $\forall i.\ \Sigma_{x\in A_i} w(x)=B$, 'no' otherwise



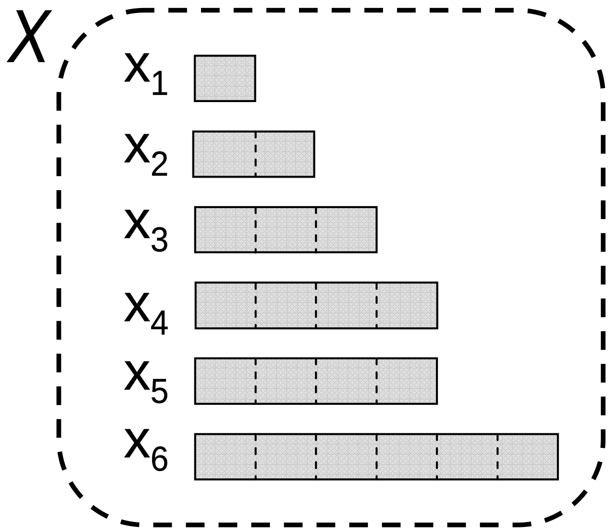
- 3-PARTITION does not have pseudo-polynomial time algorithm unless P=NP.

  (i.e., cannot be solved in poly($m+B$) unless P=NP)
- Strongly NP-complete even if $B/4 < w(x) < B/2$

An Example of Reduction (1)

An instance of 3-PARTITION P:

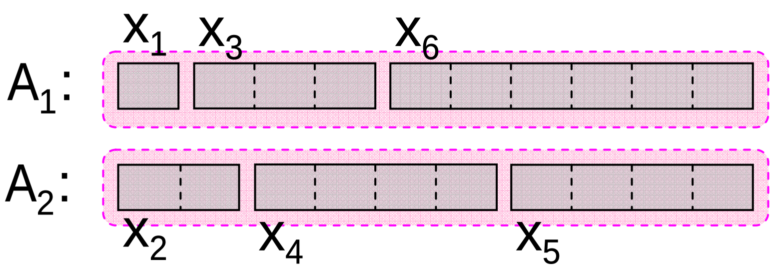
- $m=2 \rightarrow X = \{x_1,x_2,x_3,x_4,x_5,x_6\}$ ($|X|=3m=6$)
- $w=(1,2,3,4,4,6)$, $B=10$
- $w(x_1)=1$, $w(x_2)=2$, $w(x_3)=3$, $w(x_4)=4$, $w(x_5)=4$, $w(x_6)=6$



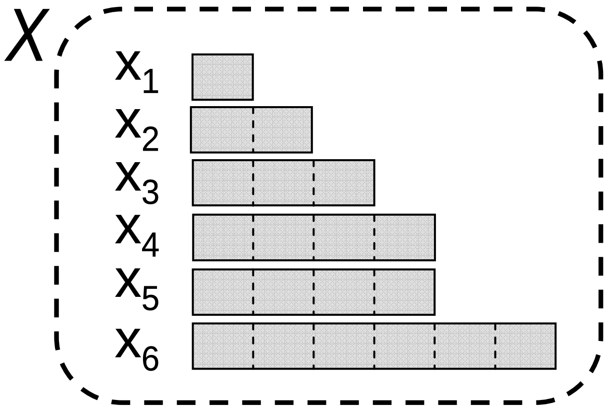
The solution for P:

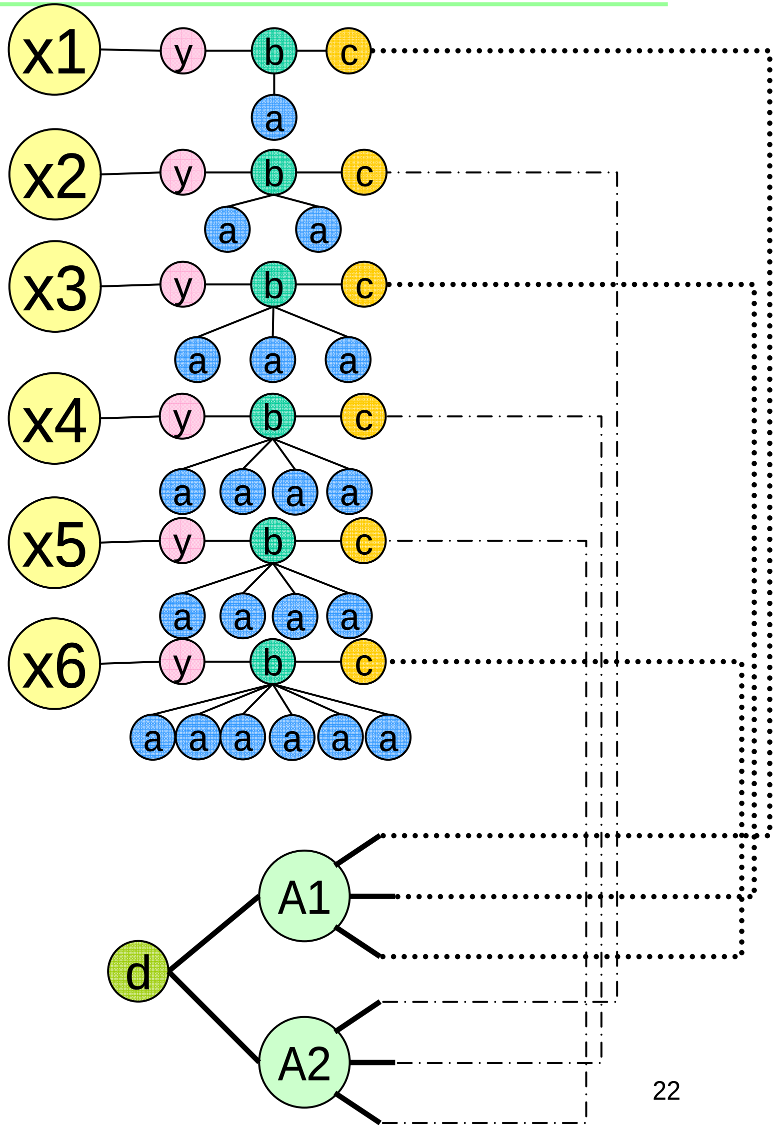
$A_1=\{x_1,x_3,x_6\} \rightarrow 1+3+6=10$

$A_2=\{x_2,x_4,x_5\} \rightarrow 2+4+4=10$

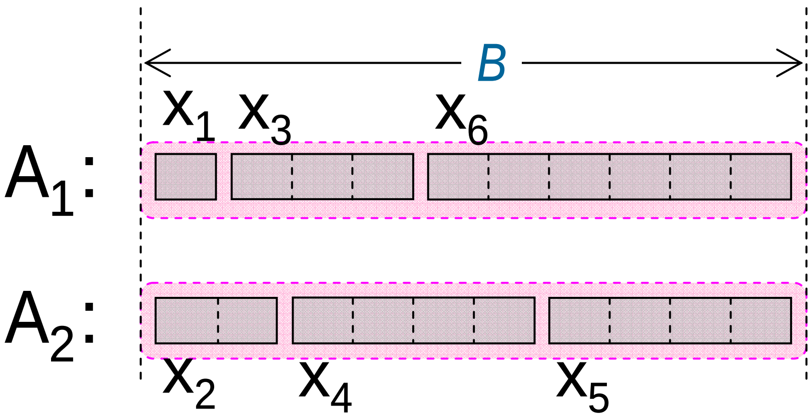


An Example of Reduction (2)





Solution for 3-PARTITION

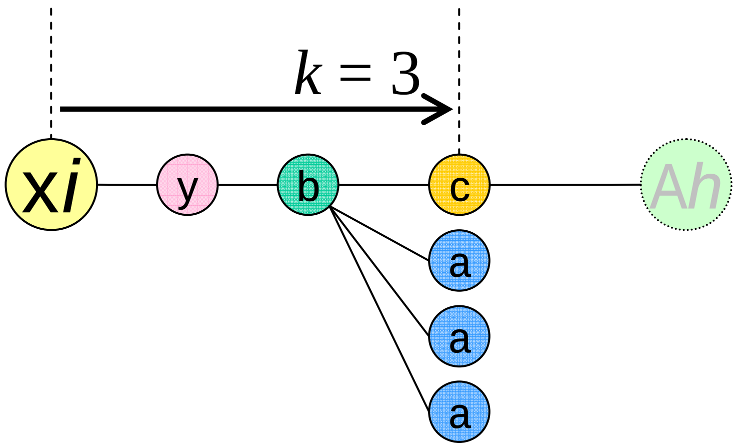


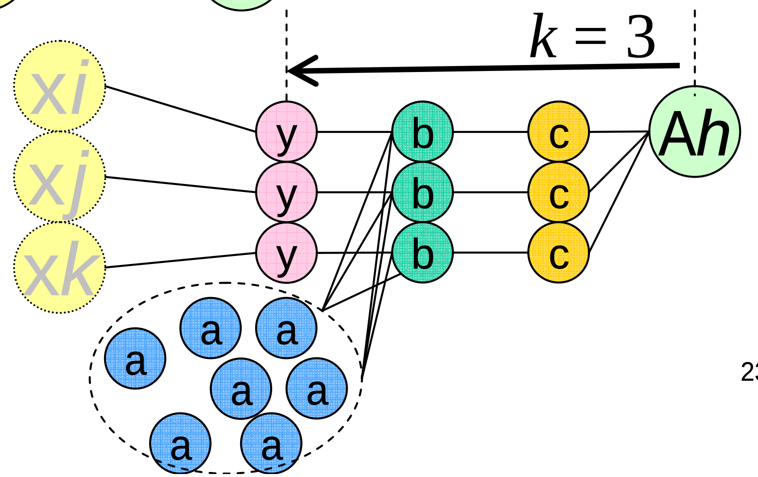
An Example of Reduction (3)

There are two kinds of vertices which have unique label. → x_i's (xi) and A_h's (Ah)

- x_i encodes the weight of i-th item → $w(x_i)$

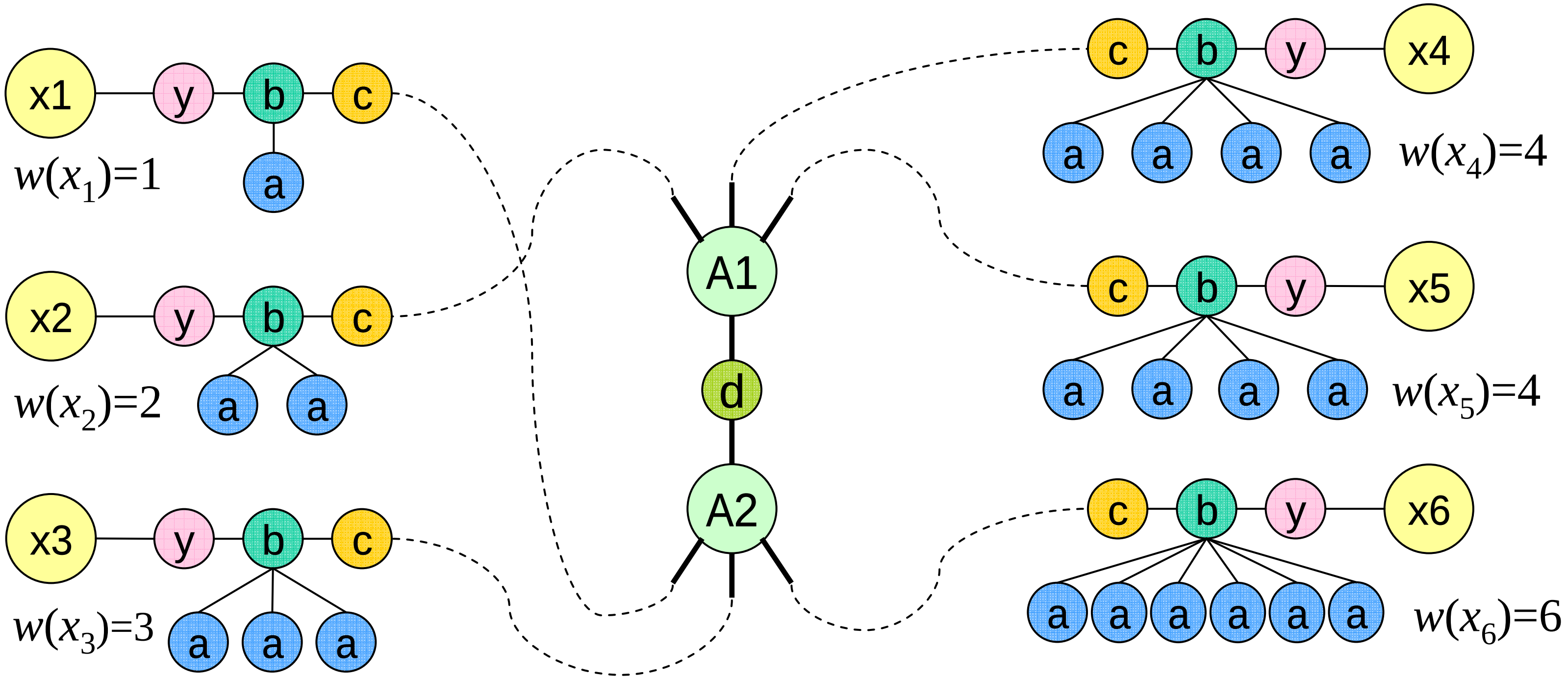
- A_h is a matchmaker of x_i's, but doesn't know who matches who, because xi and Ah are distant.





An Example of Reduction (4)

3-PARTITION *P*: *w* = (1,2,3,4,4,6) ➔ GIPF *Q*: (*v*,3)

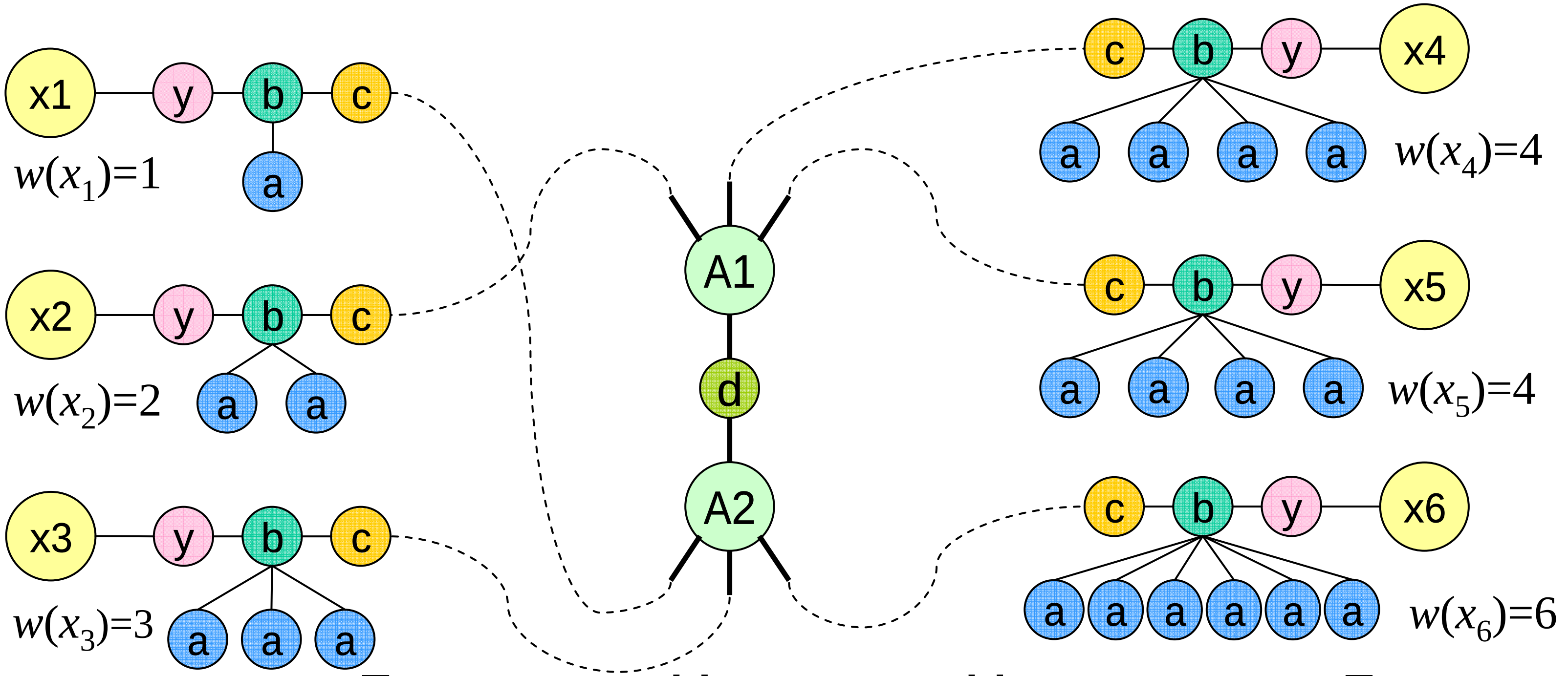


$\Sigma=\{a,b,y,c,d\}\cup\{x_1,\ldots,x_{3m}\}\cup\{A_1,\ldots,A_m\}$, $|\Sigma|=4m+5$

Feature vector specifies structures of blocks, but does not specify the connection between blocks {xi} and {A1,A2}.

An Example of Reduction (5)

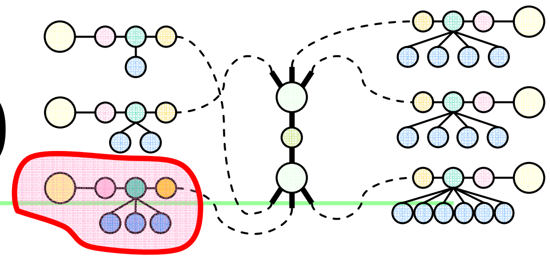
3-PARTITION *P*: *w* = (1,2,3,4,4,6) ➔ GIPF *Q*: (*v*,3)



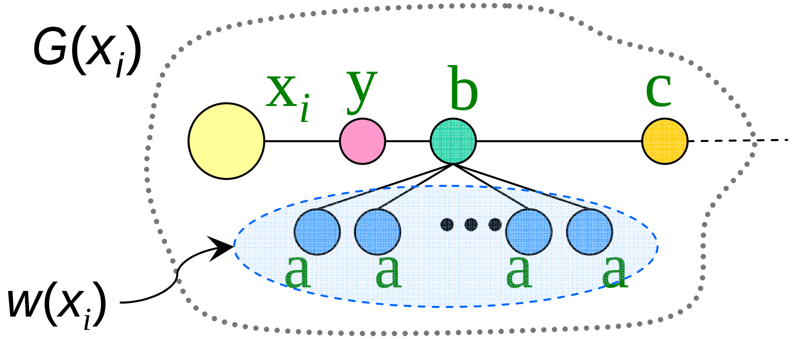
$\Sigma = \{a,b,y,c,d\} \cup \{x_1,\ldots,x_{3m}\} \cup \{A_1,\ldots,A_m\}$, $|\Sigma| = 4m+5$

- The connection satisfying the constraints given by feature vector corresponds to a solution of 3-PARTITION.
- In this case, {x1,x3,x6} and {x2,x4,x5} correspond to a solution of 3-PARTITION.

An Example of Reduction (6)



For each x_i; i=1,2,...,3m,   generate a graph $G(x_i)$:



G(x_i) encodes w(x_i)

- A label x_i is unique in the whole graph
- # of a's = w(x_i)

Paths of length ≦3 which determine G(x_i)

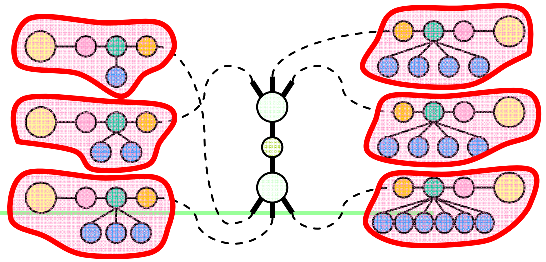
0 --- {(x_i:1), (a:w(x_i)), (y:1), (b:1), (c:1)}

1 --- {(x_iy:1), (yx_i:1), (yb:1), (by:1), (bc:1), (cb:1), (ab:w(x_i)), (ba:w(x_i))}

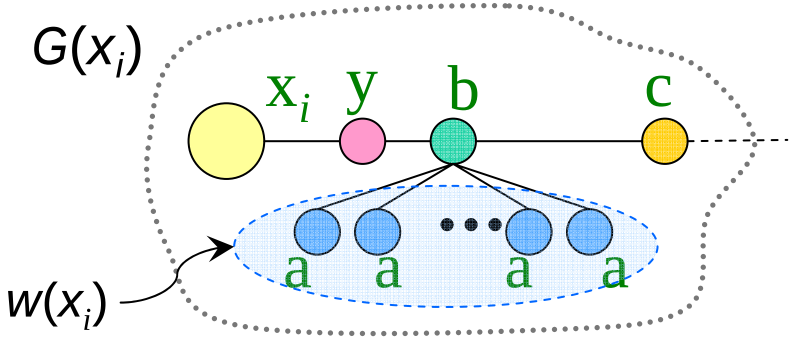
2 --- {($x_i$yb:1), (yba:w(x_i)), (ybc:1), (byx_i:1), (cby:1), (cba:w(x_i)), (aby:w(x_i)) , (abc:w(x_i)) , (aba:w(x_i)(w(x_i)-1))}

3 --- {($x_i$yba:w(x_i)), ($x_i$ybc:1), (cbyx_i:1), (abyx_i:w(x_i))}

An Example of Reduction (7)



For each x_i; i=1,2,…,3m, generate a graph $G(x_i)$:



<u>$G(x_i)$ encodes $w(x_i)$</u>

- A label x_i is unique in the whole graph
- # of a's = $w(x_i)$

<u>In total of $G(x_1)$, $G(x_2)$,…, $G(x_{3m})$</u>

0 --- $\bigcup_i\{(x_i{:}1)\} \cup \{(a{:}mB), (y{:}3m), (b{:}3m), (c{:}3m)\}$

1 --- $\bigcup_i\{(x_iy{:}1),(yx_i{:}1)\} \cup \{(ab{:}mB),(ba{:}mB),(yb{:}3m),(by{:}3m),$
$(bc{:}3m), (cb{:}3m),\}$

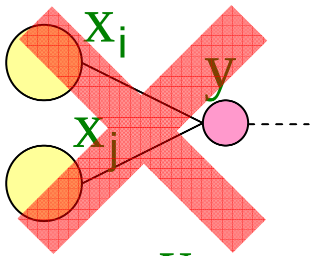
2 --- $\bigcup_i\{(x_iyb{:}1),(byx_i{:}1)\} \cup \{(yba{:}mB),(ybc{:}3m),(cby{:}3m),(cba{:}mB),$
$(aby{:}mB),(abc{:}mB),(aba{:}\ \Sigma_i w(x_i)^2\text{-}mB)\}$

3 --- $\bigcup_i\{(x_iyba{:}w(x_i)), (x_iybc{:}1), (cbyx_i{:}1), (abyx_i{:}w(x_i))\}$

Note for Uniqueness of Graph $G(x_i)$

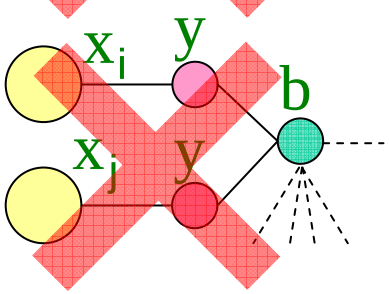
It is necessary to prove that the set of paths uniquely determines $G(x_i)$.

- The following cases does NOT occur:



--- Because a path '$x_i y x_j$' is not given

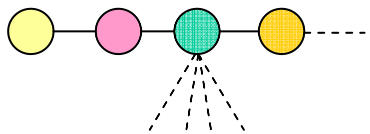
→ 1-to-1 correspondence between (x_i, y)



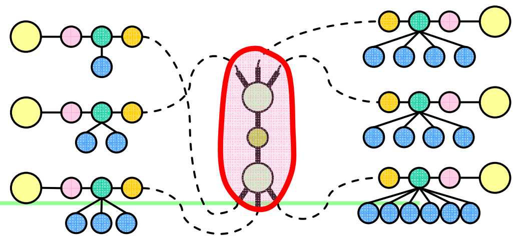
--- Because a path 'yby' is not given

(Even if tottering (backtrack) is admitted, provable similarly by using # of 'yb')

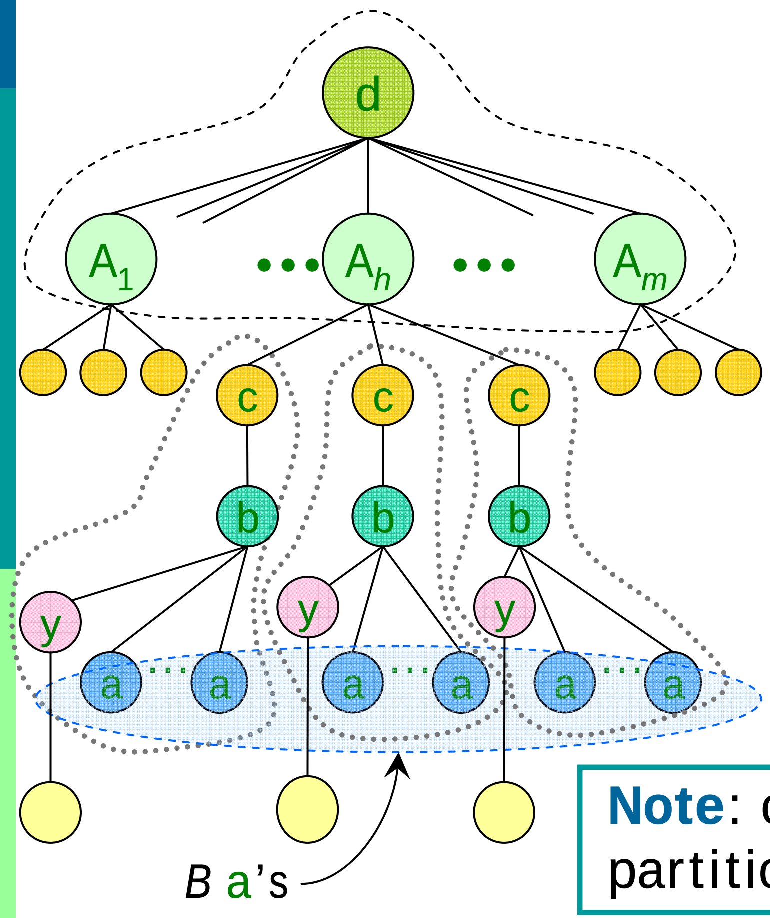
Uniqueness for a quadruple (x_i, y, b, c) can be proved in a similar way



An Example of Reduction (8)



Generation of center graph G_C



Remaing paths in G_c

0 --- $\bigcup_h\{(A_h{:}1)\} \cup \{(d{:}1)\}$

1 --- $\bigcup_h\{(A_hc{:}3m), (A_hd{:}m), (cA_h{:}3m), (dA_h{:}m)\}$

2 --- $\bigcup_h\{(A_hcb{:}3),(bcA_h{:}3),(cA_hc{:}6), (cA_hd{:}3), (dA_hc{:}3)\} \cup \bigcup_{h,k}\{(A_hdA_k{:}1)\}$

3 --- $\bigcup_h\{(A_hcby{:}3),(ybcA_h{:}3),(A_hcba{:}B), (abcA_h{:}B), (dA_hcb{:}3),(bcA_hd{:}3), (cA_hcb{:}6),(bcA_hc{:}6) \cup \bigcup_{h,k}\{(A_hdA_kc{:}3),(cA_kdA_h{:}3)\}$

Note: center graph is determined without knowing partition (without information about $w(x_i)$'s)

Strong NP-completeness of GIPF

《Theorem》 GIPF is strongly NP-complete, even if K=3 and the underlying graph is a tree.

Hardness results for other special case

《Theorem》 GIPF is strongly NP-complete, even for trees of bounded degree 4 and of fixed **Σ**.

Bounded degrees (**Δ**)

- Branchings for a's and for center d → Use binary tree

Bounded alphabets (**Σ**)

- x_i's and A_h's → Encode with fixed alphabets

In both cases, we cannot bound *K* by a constant

Note: if all of **Σ**, *K*, **Δ** are fixed, then the problem can be solved in poly. time

Conclusion

- GIPF is strongly NP-complete even if underlying graph is a tree and K=3.
- GIPF (and GIPF-M) for trees is solvable in polynomial time by using DP, if Σ, K and Δ are all fixed.

Still ongoing:

- Our DP is extendable to outer-planar graphs
- Completeness results for more restricted cases

Future work:

- Complexity of SISF-M in general cases
- Approximation algorithm, etc.