DAG パス幅に基づく種々の アルゴリズムの設計 及び DAG 木幅への拡張

2025/02/05

伊豆 真哉，川原 純

京都大学大学院情報学研究科 湊研究室

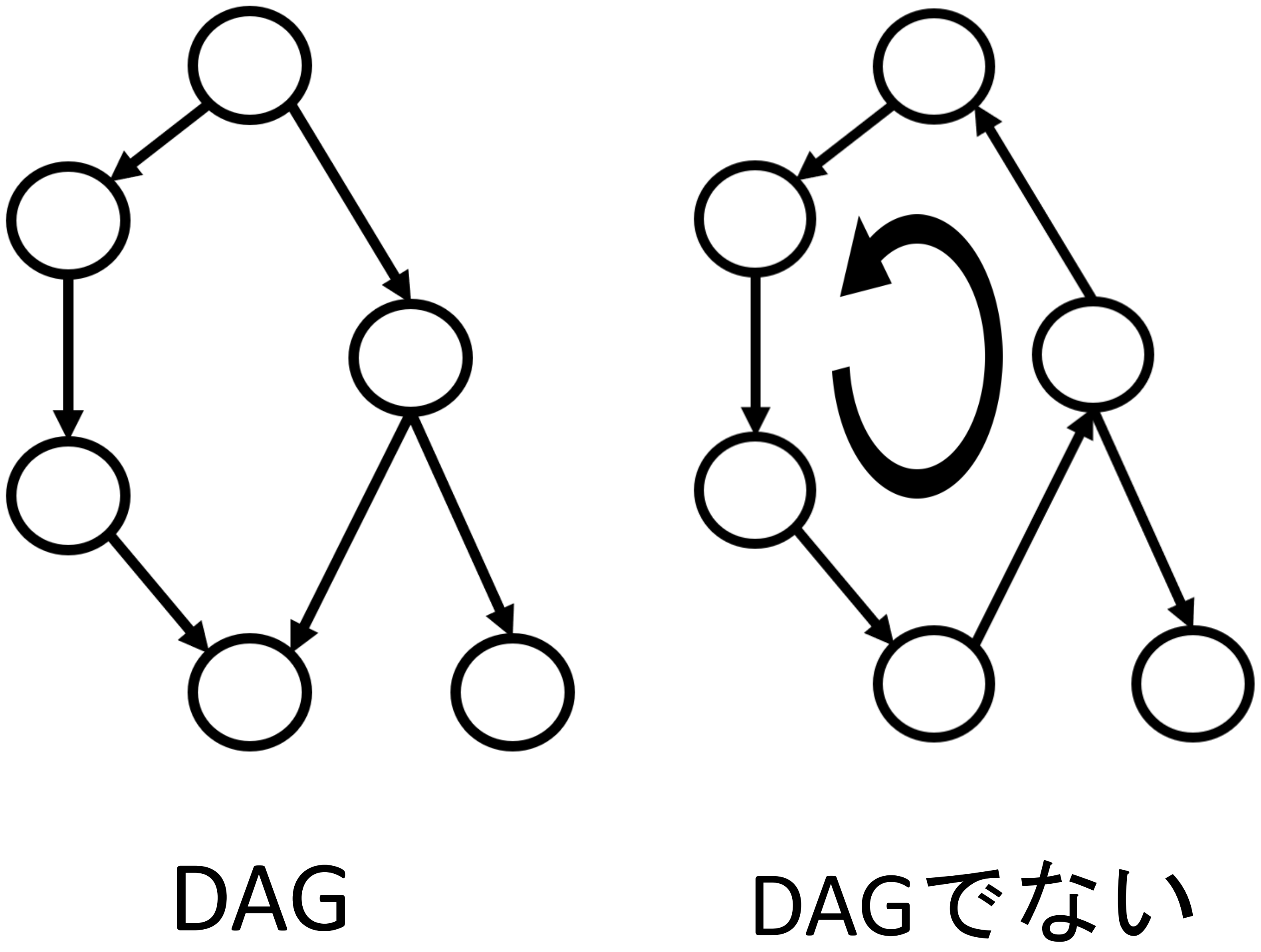
発表の流れ

1. 研究背景
2. DAGパス分解とDAGパス幅の定義
3. 有向シュタイナー木問題への応用
4. 幅の小さなDAGパス分解を求めるアルゴリズム
5. まとめ

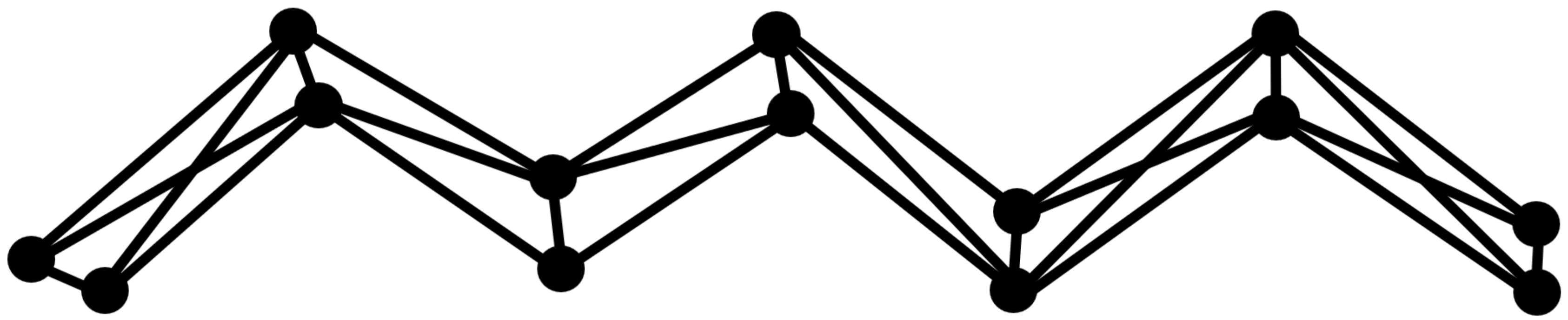
1. 研究背景

今回考えるグラフ構造【DAG】

- **DAG** (Directed Acyclic Graph)：有向非巡回グラフ．サイクルのない有向グラフ
- 今回は入力グラフがDAGの場合の組合せ問題を考える

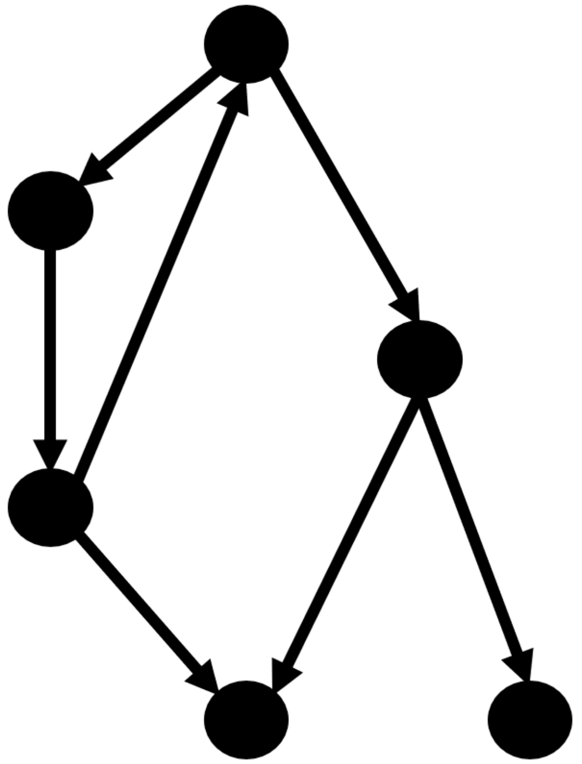


パス幅とは



- パス幅[Robert et al. 83]：
  無向グラフが**どれだけパスに近いか**

パスに近い
→ パス幅 <u>小</u>

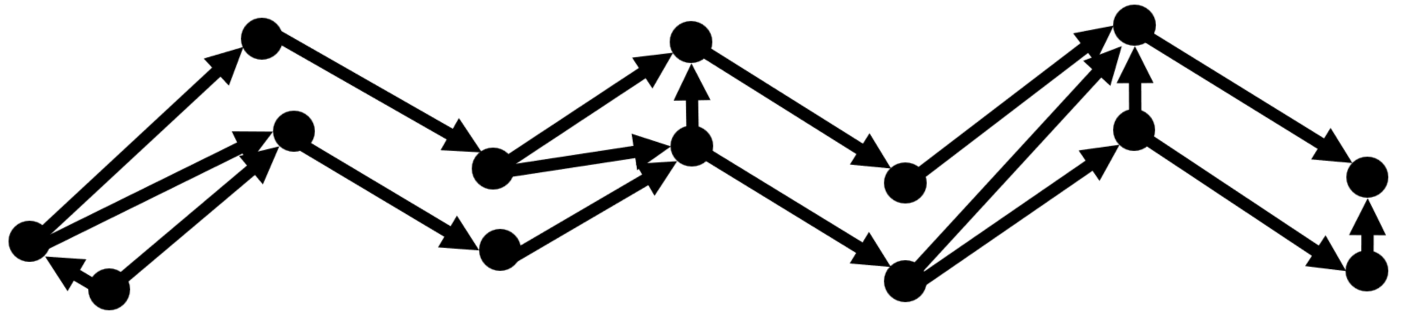


- 有向パス幅[Johnson et al. 01]：
  有向グラフが**どれだけDAGに近いか**

DAGに近い
→ 有向パス幅 <u>小</u>

- 有用性：**パラメータ化アルゴリズム**で利用
  - 最大独立集合問題　　：$2^{O(w)}n$ 時間 (n: 頂点数, w: パス幅) [Lim et al. 18]
  - 有向ハミルトン閉路問題：$n^{O(w)}$ 時間 (w: 有向パス幅) [Johnson et al. 01]

有向パス幅はDAGに対して有用でない

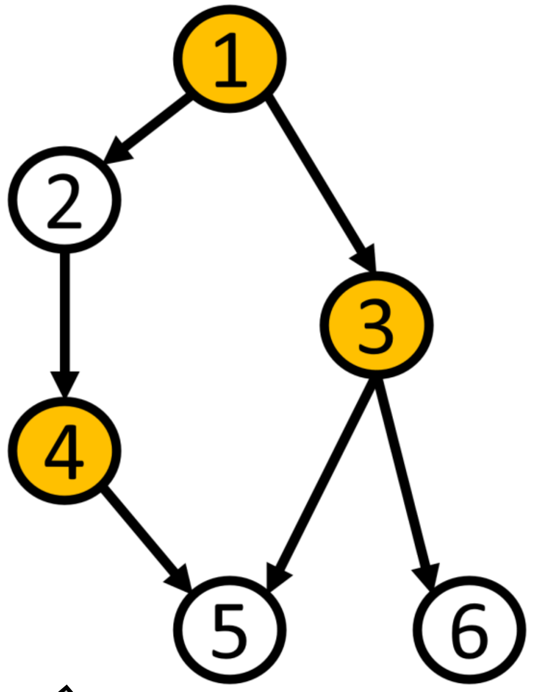


有向パスに近い → DAGパス幅 <u>小</u>

- 有向パス幅の欠点：
  入力グラフがDAGだと常に0
  (DAGはDAGへの近さ0)

DAGにも
対応させたい…

<u>有向支配集合問題</u>
(入力がDAGでもNP-hard)



- **DAGパス幅** [Kasahara et al. 23] ：
  有向グラフがどれだけ<u>有向パス</u>に近いか
  →DAGの複雑さも表現可能

有向パス幅
→解けない

DAGパス幅: w
→$O(2^w wn)$

本研究の成果【DAGパス幅に関する種々のアルゴリズムを設計】

①4つのパラメータ化アルゴリズム

有向支配集合問題→ $O(2^w wn)$

最大葉分岐数問題→ $O(2^w wn)$

k-有向点素パス問題→ $O((k+1)^w(w^2+k)\ n)$

k-有向シュタイナー木問題→ $O(2^w(k+w)n)$

n: 頂点数, w: DAGパス幅

②DAGパス幅を求める近似アルゴリズム

$\boldsymbol{O(\log^2 n)}$**-近似**

→セパレータを利用

$\boldsymbol{O(\log^{3/2} n)}$**-近似**

→Pebbling gameを利用

③nによらない幅を求めるアルゴリズム

幅が高々$\boldsymbol{O(ld^k)}$のDAGパス分解

→グラフの埋め込みを利用

※l: 根数, d: 最大出次数, k: 入力整数

④DAG木幅への拡張

DAG木幅→ 有向木への近さを表現

→ DAGパス幅より小

有向支配集合問題→ $O(2^w w^2 n)$

本研究の成果【DAGパス幅に関する種々のアルゴリズムを設計】

①4つのパラメータ化アルゴリズム

有向支配集合問題→ $O(2^w wn)$

最大葉分岐数問題→ $O(2^w wn$

k-有向点素パス問題→ $O((k + 1$ $)$

k-有向シュタイナー木問題→ $O(2^w(k + w)n)$

今回説明

n: 頂点数, w: DAGパス幅

③nによらない幅を求めるアルゴリズム

今回説明

幅が高々$\boldsymbol{O(ld^k)}$のDAGパス分解

→グラフの埋め込みを利用

※l: 根数, d: 最大出次数, k: 入力整数

②DAGパス幅を求める近似アルゴリズム

$O(\log^2 n)$-近似

→セパレータを利用

$O(\log^{3/2} n)$-近似

→Pebbling gameを利用

④DAG木幅への拡張

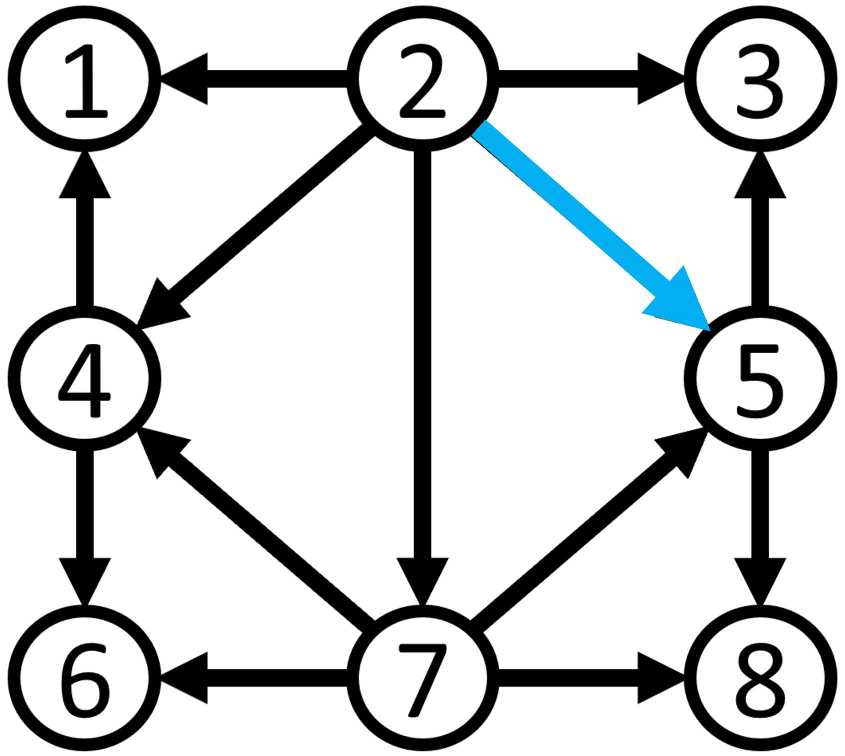
DAG木幅→ 有向木への近さを表現

→ DAGパス幅より小

有向支配集合問題→ $O(2^w w^2 n)$

2. DAGパス分解とDAGパス幅の定義

DAGパス分解・DAGパス幅 [Kasahara et al. 23]



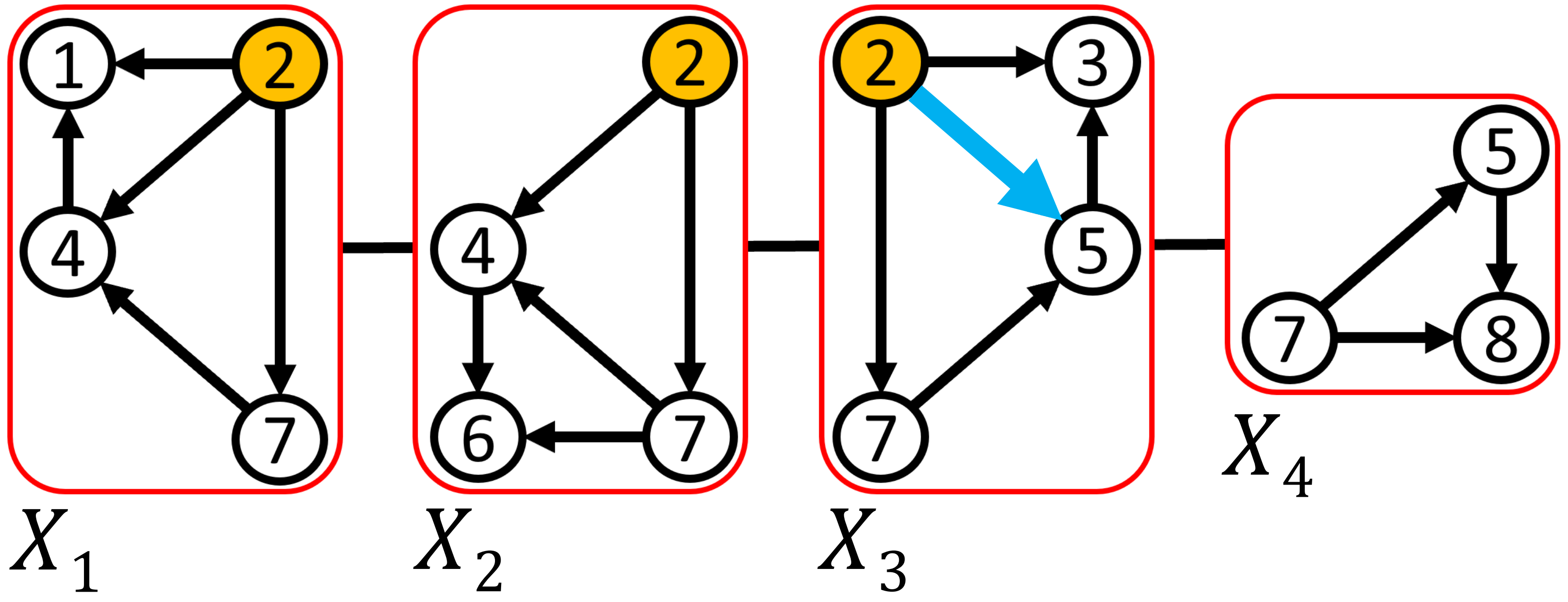
■ **DAGパス分解**：有向グラフ$G = (V, E)$に対し，以下を満たすパス$X = (X_1, X_2, \ldots, X_s)$.

1. $X_1 \cup X_2 \cup \cdots \cup X_s = V$
2. 各$(u, v) \in E$に対し，あるiがあり，$u, v \in X_i$, $\boldsymbol{v \notin X_{i-1}}$
3. 各$v \in V$に対し，vを含むバッグは連結なパスを構成する

パス分解との違い

DAGパス分解

幅=DAGパス幅=3



■ 幅：$\max_{1 \le i \le s} |X_i|$ -1

■ **DAGパス幅**：最小の幅を与えるDAGパス分解の幅. 値が小さいほど**有向パスに近い**グラフ

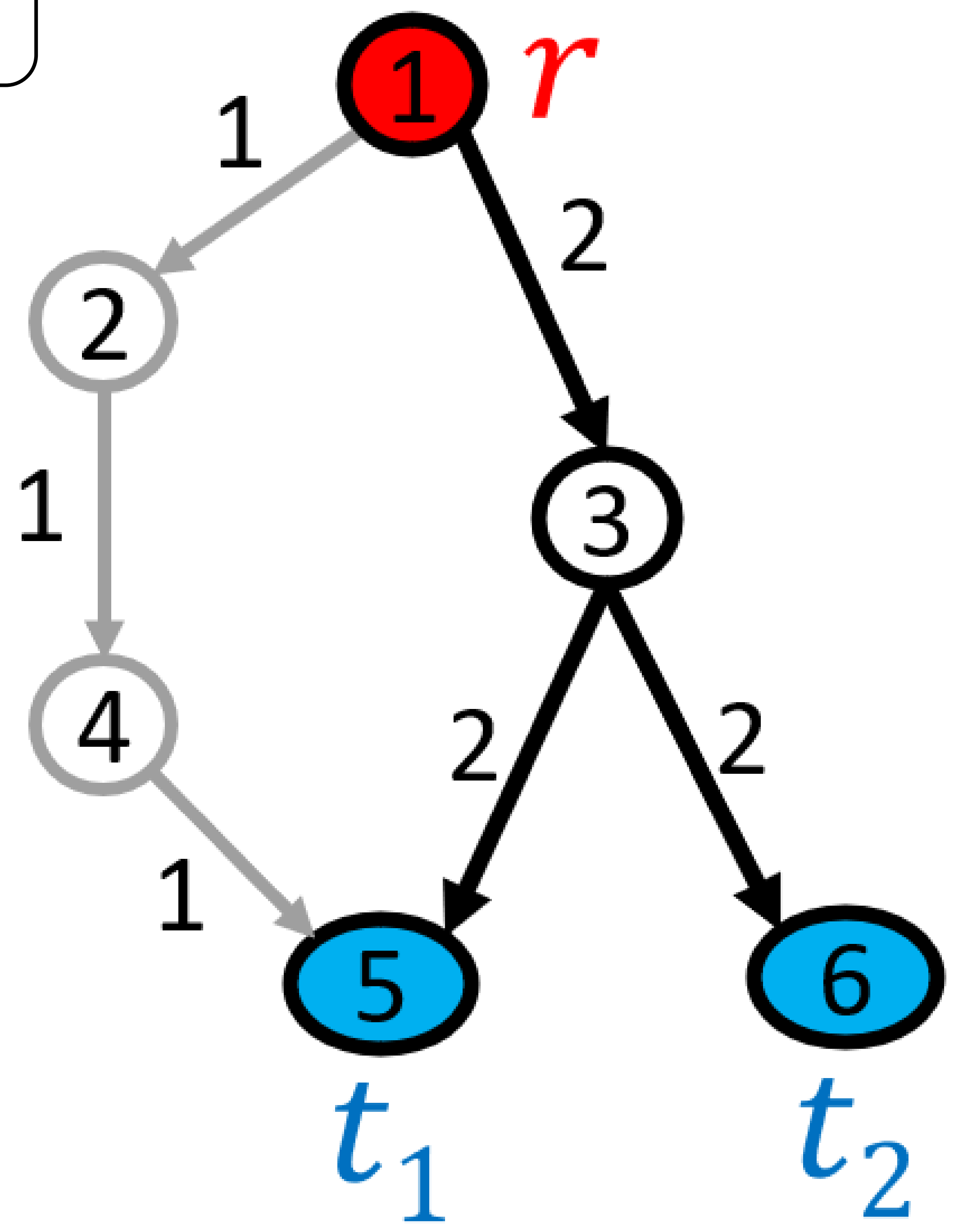
成果①

3. 有向シュタイナー木問題への応用

有向シュタイナー木問題

今回はDAG上

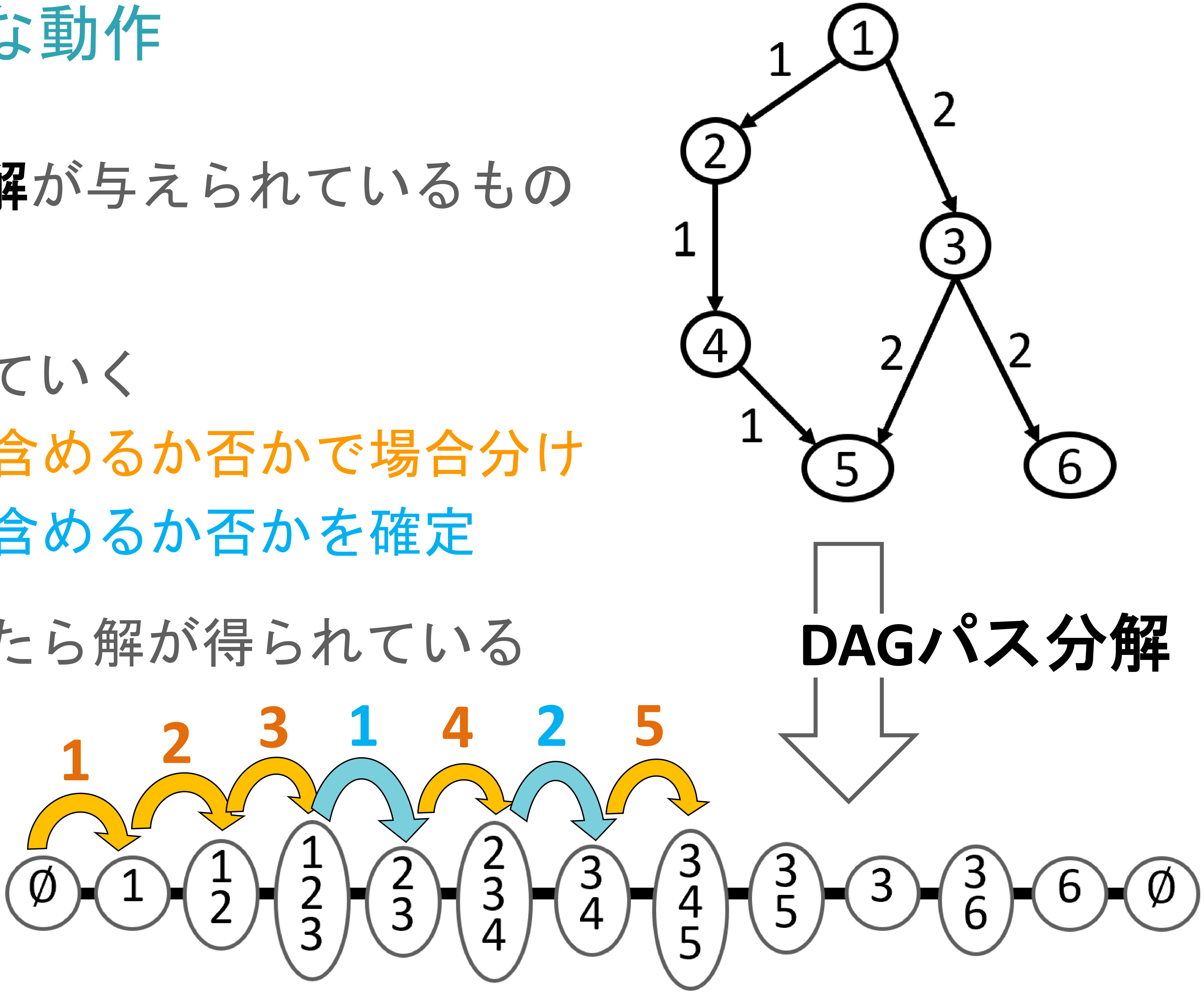
- input：枝重み付き有向グラフ$G = (V, E)$,
  $r \in V, R = \{t_1, t_2, \ldots, t_k\} \subseteq V$
- objective：rを根とし，Rを含む有向木の総枝重みの最小値
- DAG上でもNP-hard [Ganian et.al 14]
- 無向パス幅を使ったFPTの先行研究は(調べた限り)行われていない



総枝重み最小のDSTの例
(総枝重み6)

アルゴリズムの大まかな動作

- 入力グラフの**DAGパス分解**が与えられているものとする
- バッグを左側から順に見ていく
  - 頂点vが追加→vを解に含めるか否かで場合分け
  - 頂点vが削除→vを解に含めるか否かを確定
- 右端のバッグまで到達したら解が得られている



具体的な計算式

$ST(i; A_i, B_i)$：今考えているバッグX_iでの最適解

頂点vが追加

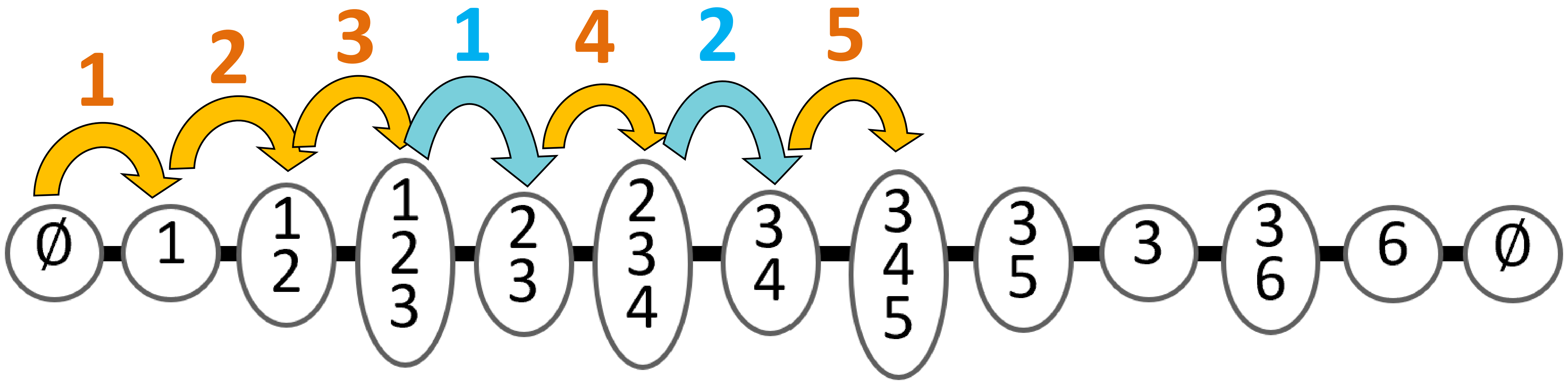
$$ST(i; A_i, B_i) = \begin{cases} ST(i-1; A_i \setminus \{v\}, B_i) + \min\limits_{u \in pred(v) \cap A_i} d(u,v) & (v \in A_i \text{かつ} pred(v) \cap A_i \neq \emptyset) \\ ST(i-1; A_i, B_i \setminus \{v\}) & (v \in B_i \text{かつ} v \notin R \cup \{r\}) \\ \infty & (otherwise) \end{cases}$$

頂点vが削除

$$ST(i; A_i, B_i) = \min\{ST(i-1; A_i \cup \{v\}, B_i), ST(i-1; A_i, B_i \cup \{v\})\}$$

- 計算量 (w: DAGパス幅)

→ $\boldsymbol{O(2^w(w + |R|)n + n^2)}$



本アルゴリズムの利点と問題点

■ **利点**：

DAGパス分解が矢印の向きに沿って構成

→根から葉に向かって有向木を構築しやすい

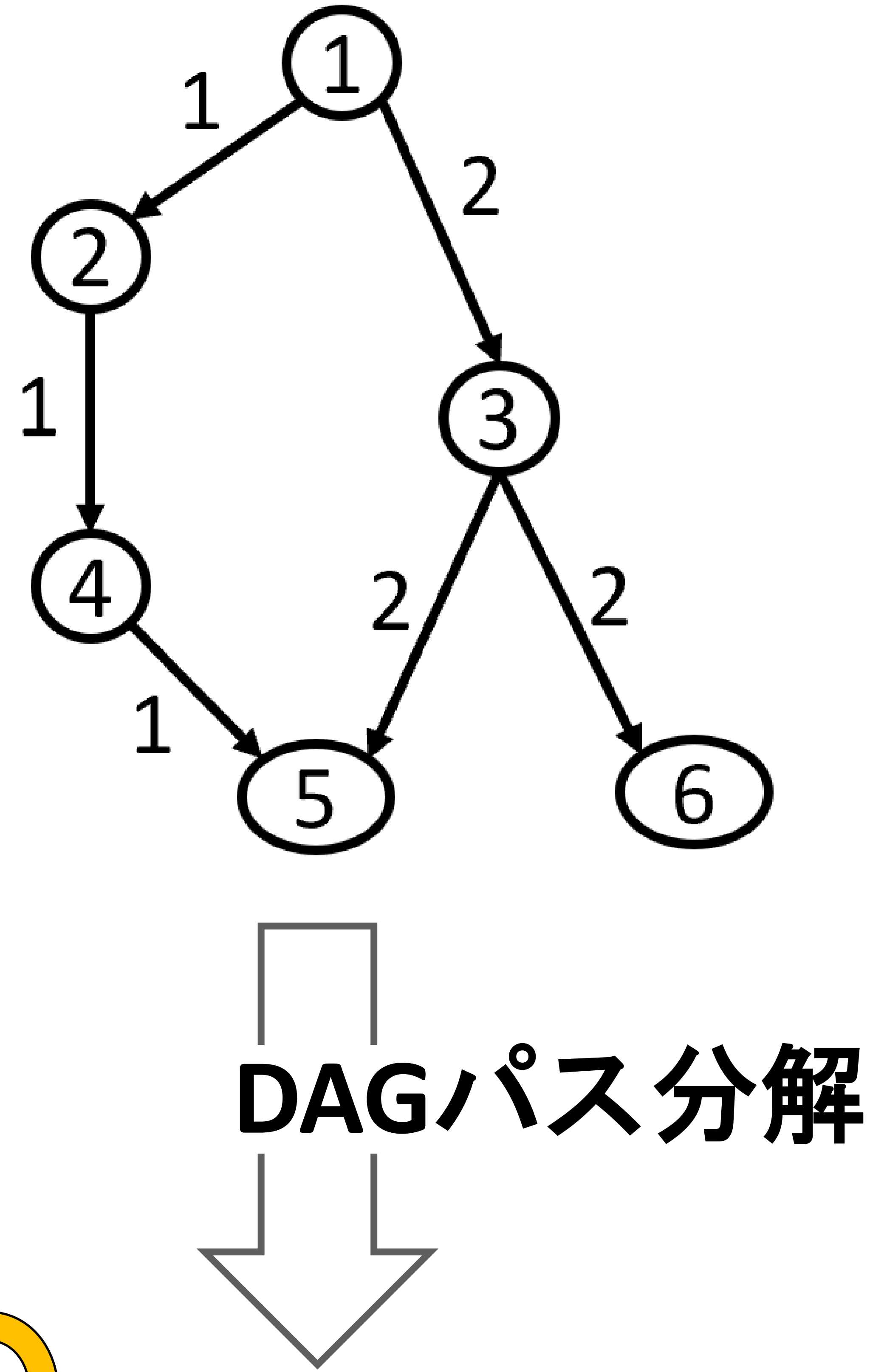
→アルゴリズムが単純に

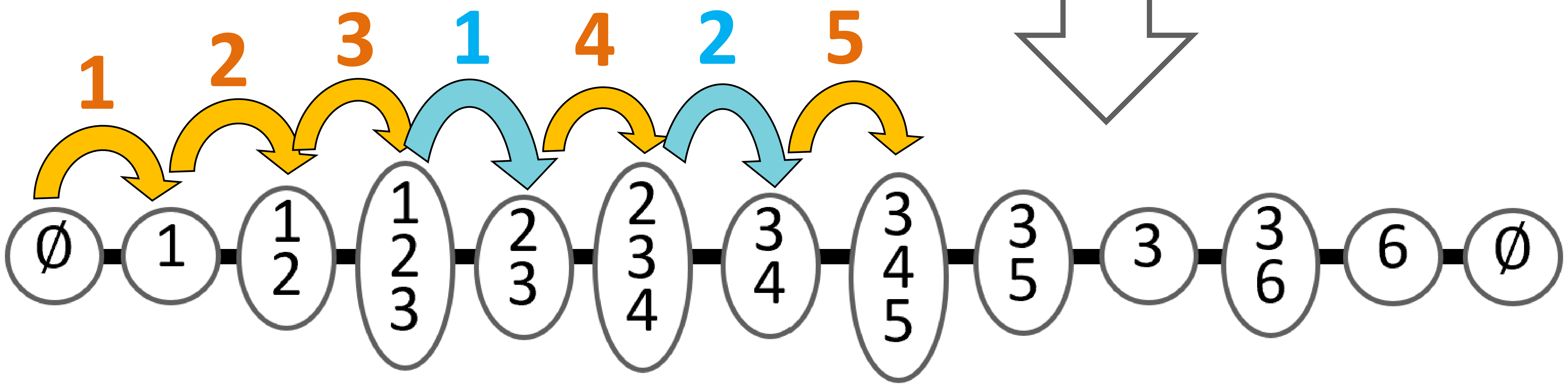
■ **問題点**：

あらかじめ**幅の小さなDAGパス分解**が必要

→実は構築が難しい

→**成果③で構築**





成果③

4. 幅の小さなDAGパス分解を求めるアルゴリズム

パス幅の計算は難しい

- 入力整数kに対し，Gのパス幅はk以下か？→ **NP-complete**
- ではどうするか…→　<u>**以下のアルゴリズム**</u>を利用する
- DAGパス幅のアルゴリズムは知られていなかった

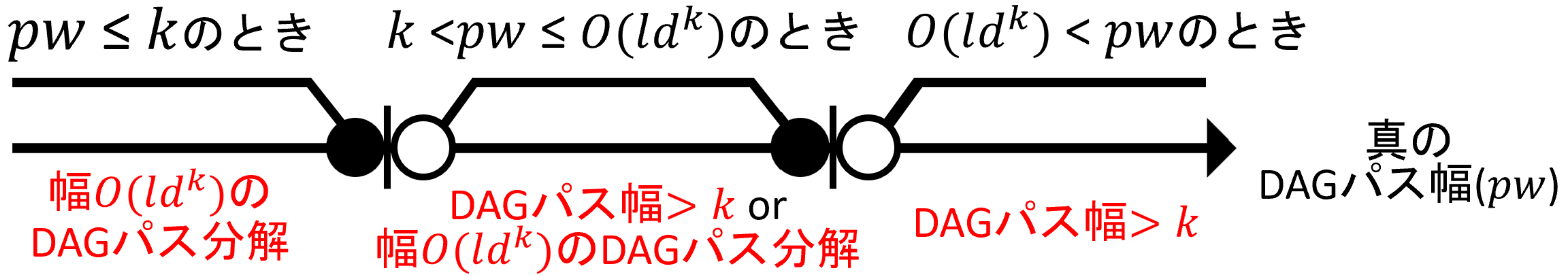
入力：頂点数$\boldsymbol{n}$のグラフ$\boldsymbol{G}$, 整数$\boldsymbol{k}$
出力：「パス幅 > $\boldsymbol{k}$の事実と証拠」or「幅が高々$\boldsymbol{w(k)}$のパス分解」
時間：$\boldsymbol{f(n,k)}$　(nに対し多項式時間)

パス幅の種類	G	$w(k)$	$f(n,k)$	参照
パス幅	無向グラフ	$O(2^k)$	$O(2^k n)$	[Kevin et al. 96]
		k	$2^{O(k^3)}n$	[Bodlaender 96]
有向パス幅	有向グラフ	k	$O(mn^{k+1})$　(m=$\|E(G)\|$)	[Tamaki 2011]
DAGパス幅	***open***			

本研究の成果③

- 「DAGパス幅 > kの事実と証拠」or「幅が高々$O(ld^k)$のDAGパス分解」のいずれか1つを出力するアルゴリズムを初めて構築

(l: Gの根数, d: Gの最大出次数)

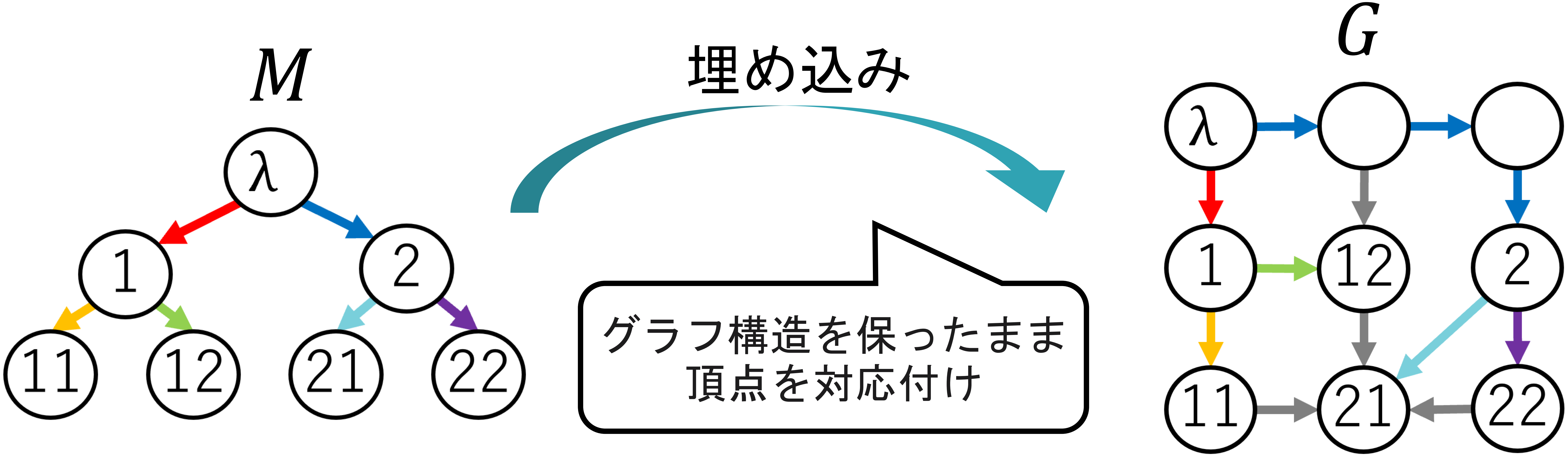


パス幅の種類	G	$w(k)$	$f(k)$	参照
パス幅	無向グラフ	$O(2^k)$	$O(2^k n)$	[Kevin et al. 96]
		k	$2^{O(k^3)} n$	[Bodlaender 96]
有向パス幅	有向グラフ	k	$O(mn^{k+1})$ ($m=\|E(G)\|$)	[Tamaki 2011]
DAGパス幅	DAG	$\boldsymbol{O(ld^k)}$	$\boldsymbol{O(d^k n^2)}$	

参考

アルゴリズム構築の準備

- **埋め込み** (DAG $\boldsymbol{M}$ → DAG $\boldsymbol{G}$)：
  - $\boldsymbol{M}$の頂点をグラフ構造を保ったまま$\boldsymbol{G}$の各頂点に対応付ける
- 埋め込み可能ならば…
  - $\boldsymbol{G}$のDAGパス幅 ≥ $\boldsymbol{M}$のDAGパス幅

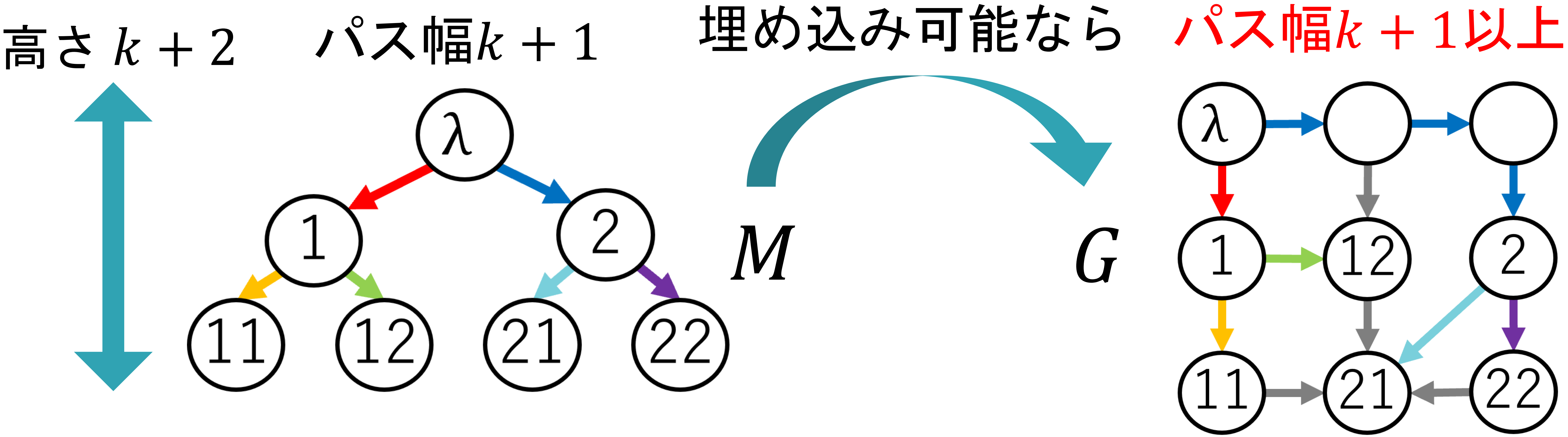


アルゴリズムのアイデア

- 入力：整数$\boldsymbol{k}$, DAG $\boldsymbol{G}$ (最大出次数$\boldsymbol{d}$)
- $\boldsymbol{M}$：出次数d, 高さ$k+2$の完全有向木

DAGパス幅 = $\boldsymbol{k+1}$

- $(\boldsymbol{M} \to \boldsymbol{G})$に埋め込み可能ならば…
  - $\boldsymbol{G}$のDAGパス幅 ≥ $\boldsymbol{M}$のDAGパス幅 $(= \boldsymbol{k+1})$

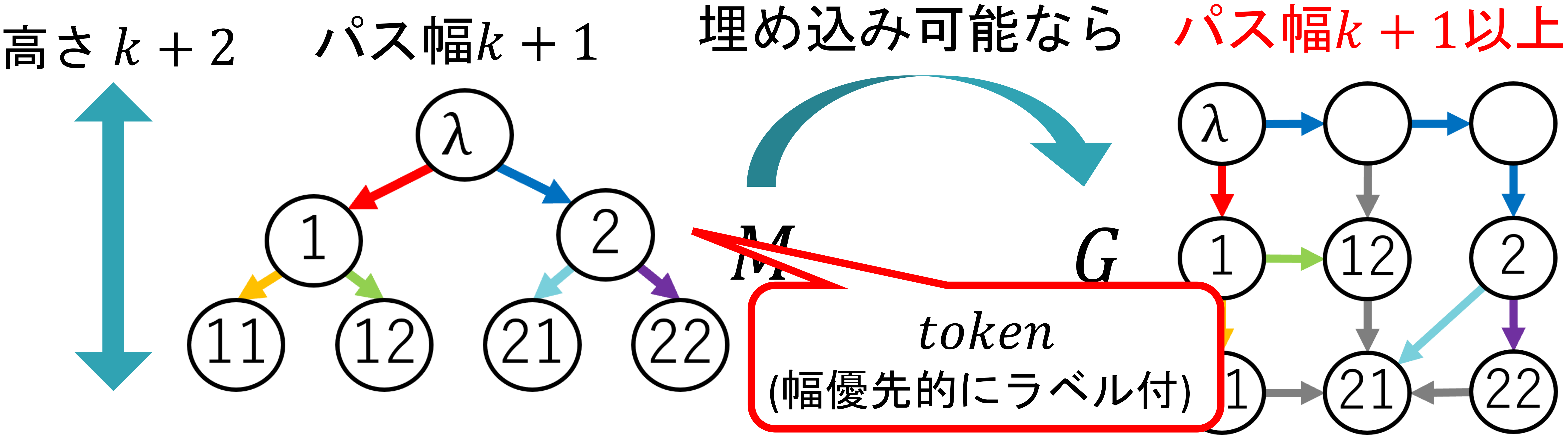


アルゴリズムのアイデア

- 入力：整数$\boldsymbol{k}$, DAG $\boldsymbol{G}$ (最大出次数$\boldsymbol{d}$)
- $\boldsymbol{M}$：出次数d, 高さ$k+2$の完全有向木

DAGパス幅 = $\boldsymbol{k}+\mathbf{1}$

- $(\boldsymbol{M} \to \boldsymbol{G})$に埋め込み可能ならば…
  - $\boldsymbol{G}$のDAGパス幅 $\geq$ $\boldsymbol{M}$のDAGパス幅 $(= \boldsymbol{k}+\mathbf{1})$

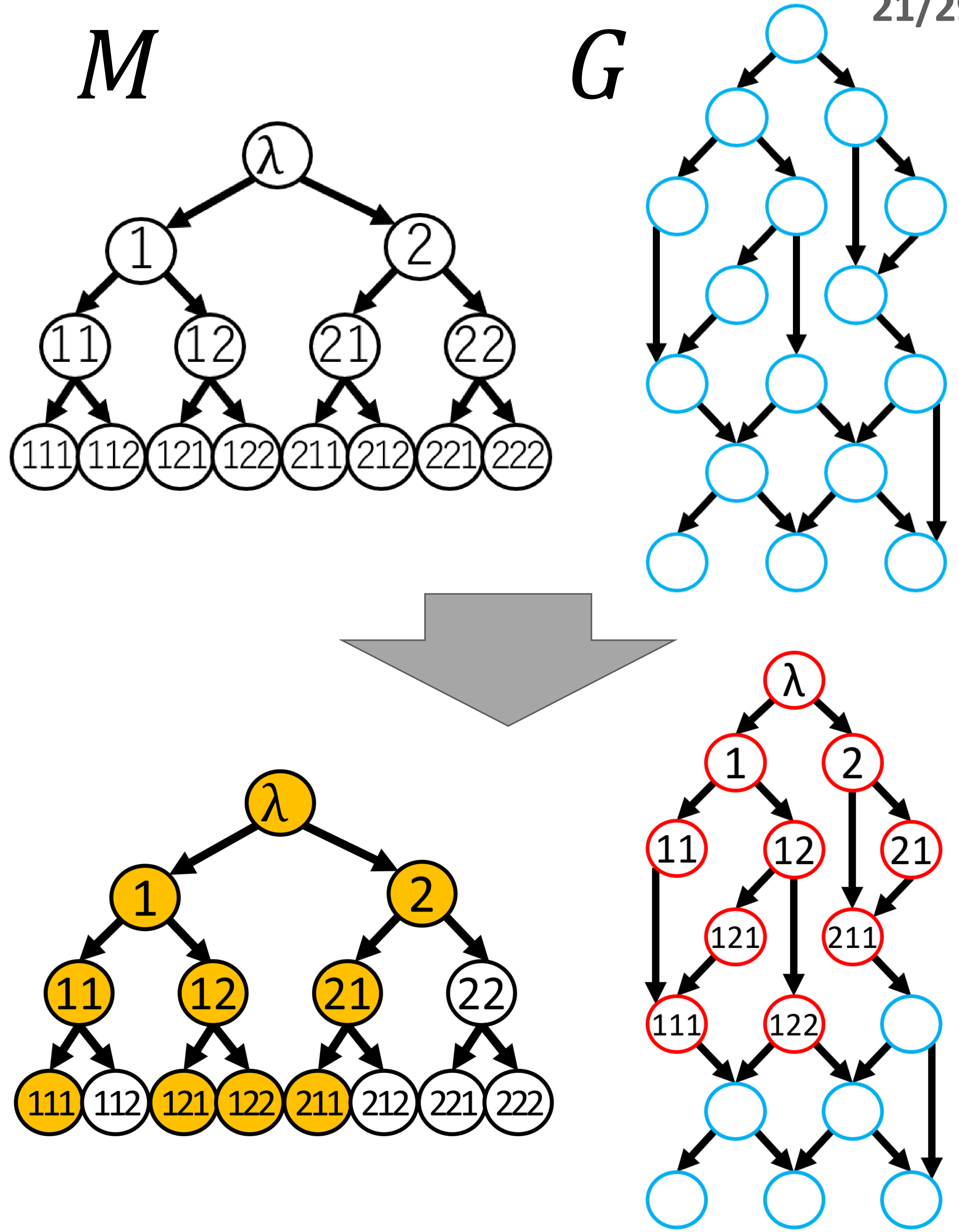


アルゴリズム1：*GrowTokenTree*

- 初めにGの全頂点はblue
  - → *token*が置かれたらredに
  - → *token*が除かれてもredのまま

GrowTokenTree

- Mの木構造を保ったままgreedyにGに*token*を(可能な限り)置いていく
- 親全てに*token*が置かれている場合のみ自身に*token*を置ける
- 最後に*token*を置いた頂点集合のみ出力



アルゴリズム2：*FindEmbedding*

1. Gの根に$token\ \lambda$を置く
2. $X_1 \leftarrow GrowTokenTree$. これをDAGパス分解の**<u>最初のバッグ</u>**
3. until 【Gの全頂点がred】or【$|X_i| = |V[M]|$】:

   if {ある$token\ T\ (on\ v \in\ V[G])$があり, $suc(v)$が全てred, かつTの配置済の子$token$が高々1個}:

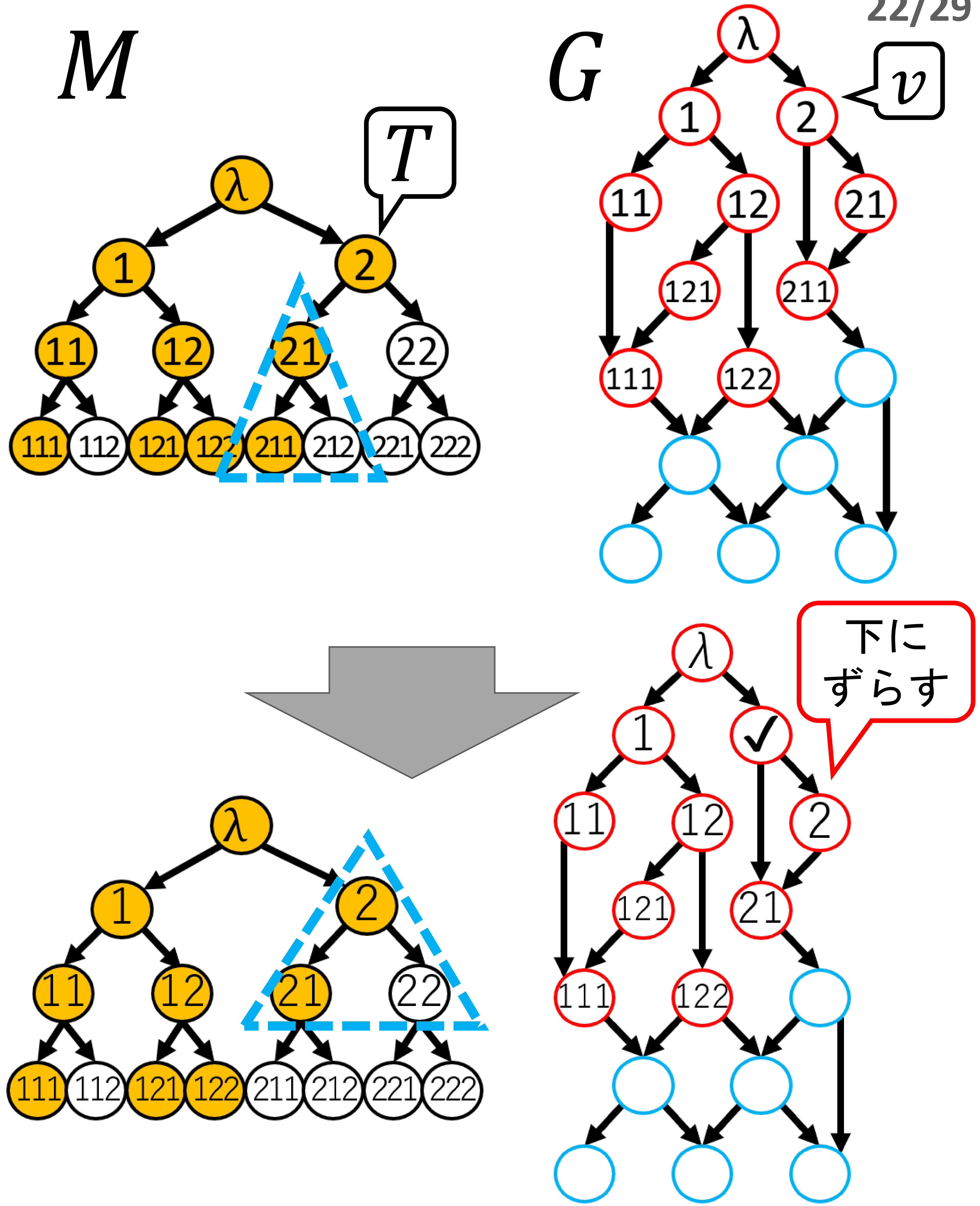
   - Tをvから取り除く
   - Tが配置済の子$T \cdot b$をもつ: $T \cdot b \cdot S \leftrightarrow T \cdot S$

   else:

   - return X_i

   $X_{i+1} \leftarrow GrowTokenTree$. これを**<u>i+1番目のバッグ</u>**

*token T*を選んで
“下にずらしていく”



■ $GrowTokenTree$の出力がDAGパス分解の各バッグに対応

①②③のいずれかで終了

1. Gの根に$token\ \lambda$を置く
2. $X_1 \leftarrow GrowTokenTree$. これをDAGパス分解の最初のバッグ
3. until 【Gの全頂点がred】 or 【$|X_i| = |V[M]|$】 : 2

   1 if {ある$token\ T$ (on $v \in V[G]$)があり, $suc(v)$が全てred, かつTの配置済の子$token$が高々1個}:

   ・Tをvから取り除く

   ・Tが配置済の子$T \cdot b$をもつ: $T \cdot b \cdot S \leftrightarrow T \cdot S$

   else:

   ・return X_i 3

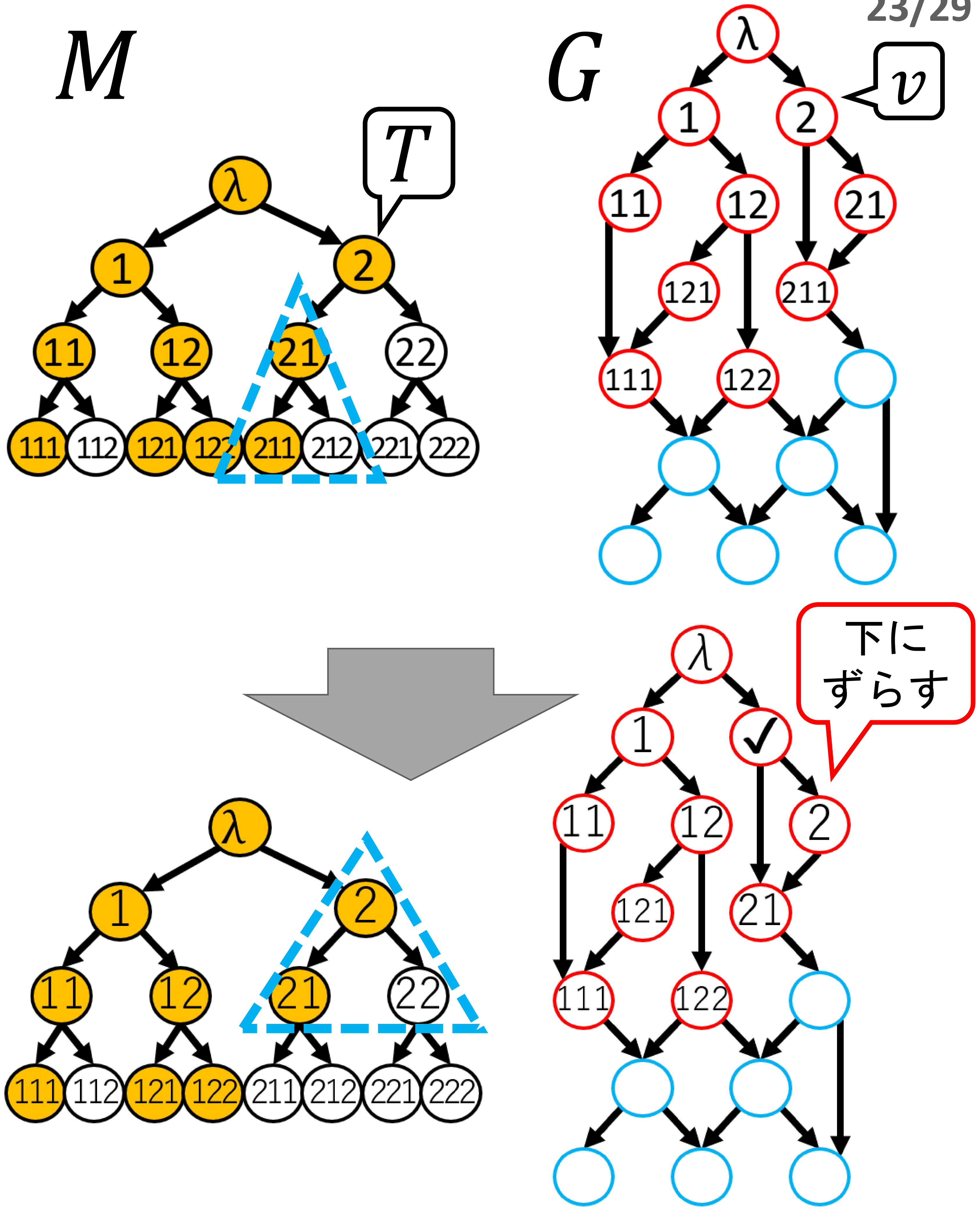
   $X_{i+1} \leftarrow GrowTokenTree$. これを$i$+1番目のバッグ

■ $i = s$で終了したとき

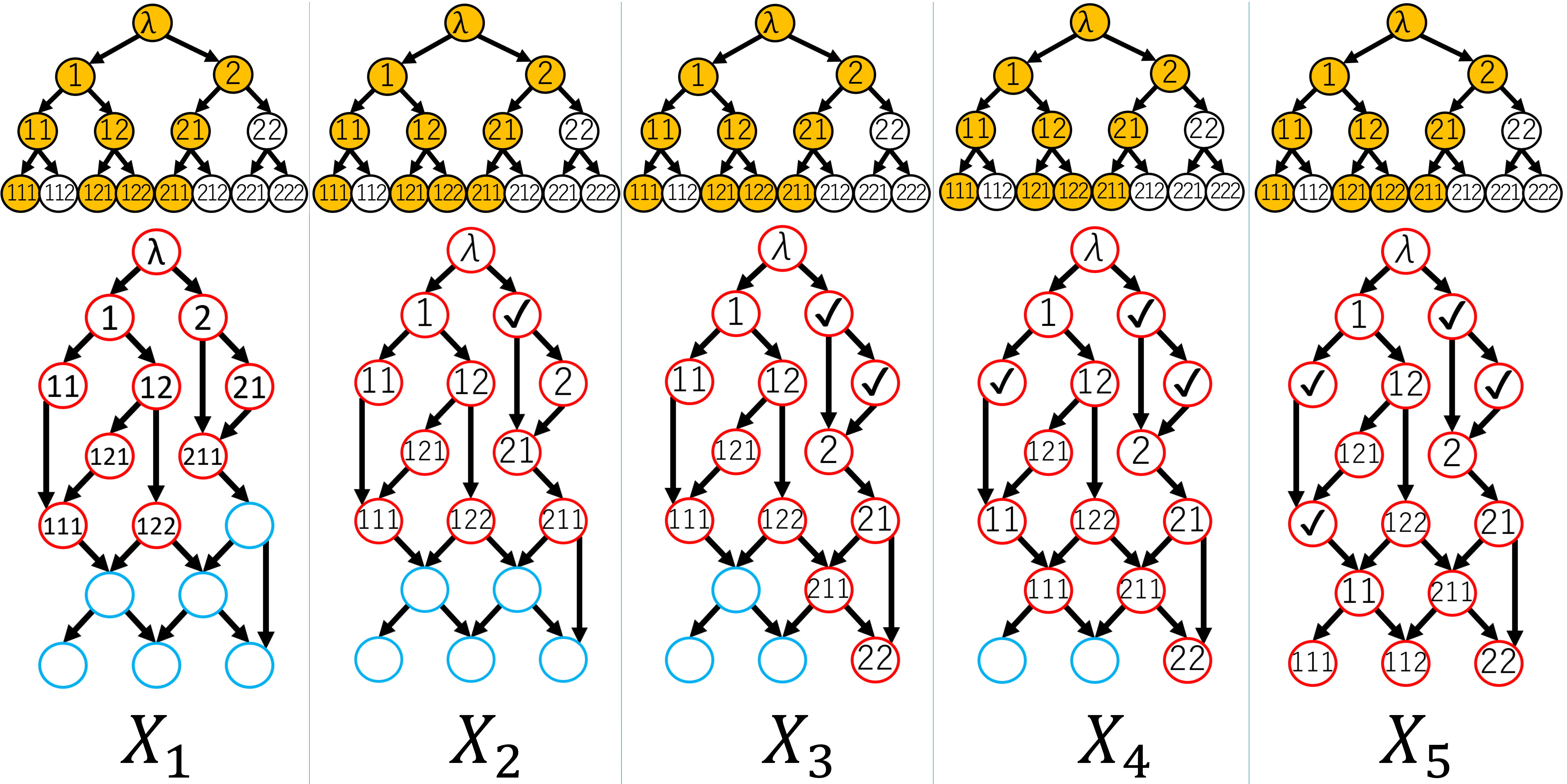
① → $(X_1, X_2, \ldots, X_s)$が幅$O(ld^k)$のDAGパス分解

② → DAGパス幅 $> k$

③ → DAGパス幅 $> k$



アルゴリズムの動作例 (①幅$O(ld^k)$のDAGパス分解を出力)



先行研究[Kevin et al. 96]との違い

[Kevin et al. 96]

1. Gの任意の頂点に$token\ \lambda$を置く
2. $i=1$とし，X_i ←call $GrowTokenTree$
3. until【Gの全頂点がred】or【$|X_i|=|V[M]|$】:
   - $token\ T\ (on\ v\in V[G'],$ 配置済の子$token$が高々1個$)$を選択
   - Tをvから取り除く
   - Tが配置済の子$T\cdot b$をもつ: $T\cdot b\cdot S \leftrightarrow T\cdot S$
   - $i=i+1$とし, X_i ←call $GrowTokenTree$

このようなTが**必ず**存在

提案アルゴリズム

1. Gの根に$token\ \lambda$を置く
2. $i=1$とし，X_i ←call $GrowTokenTree$
3. until【Gの全頂点がred】or【$|X_i|=|V[M]|$】:
   if {ある$token\ T\ (on\ v\in V[G'])$があり, $suc(v)$が全てred, かつTの配置済の子$token$が高々1個}:
   - Tをvから取り除く
   - Tが配置済の子$T\cdot b$をもつ: $T\cdot b\cdot S \leftrightarrow T\cdot S$

   else:
   - return X_i

   3

   $i=i+1$とし，X_i ←call $GrowTokenTree$

Tが存在する**とは限らない**

- 有向グラフの埋め込みは**無向グラフよりも難しい**
- $token\ T$が見つからず③で終了しても「***G*の DAGパス幅 > *k***」を示した

証明の方針

①：幅$O(ld^k)$のDAGパス分解を出力

→ X_iの列がDAGパス分解の3つのルールを満たし，かつ高々$|V[M]|$個の$token$のみ使う

②：GのDAGパス幅$> k$

→ X_sがMからGへの埋め込みになっている

③： GのDAGパス幅$> k$

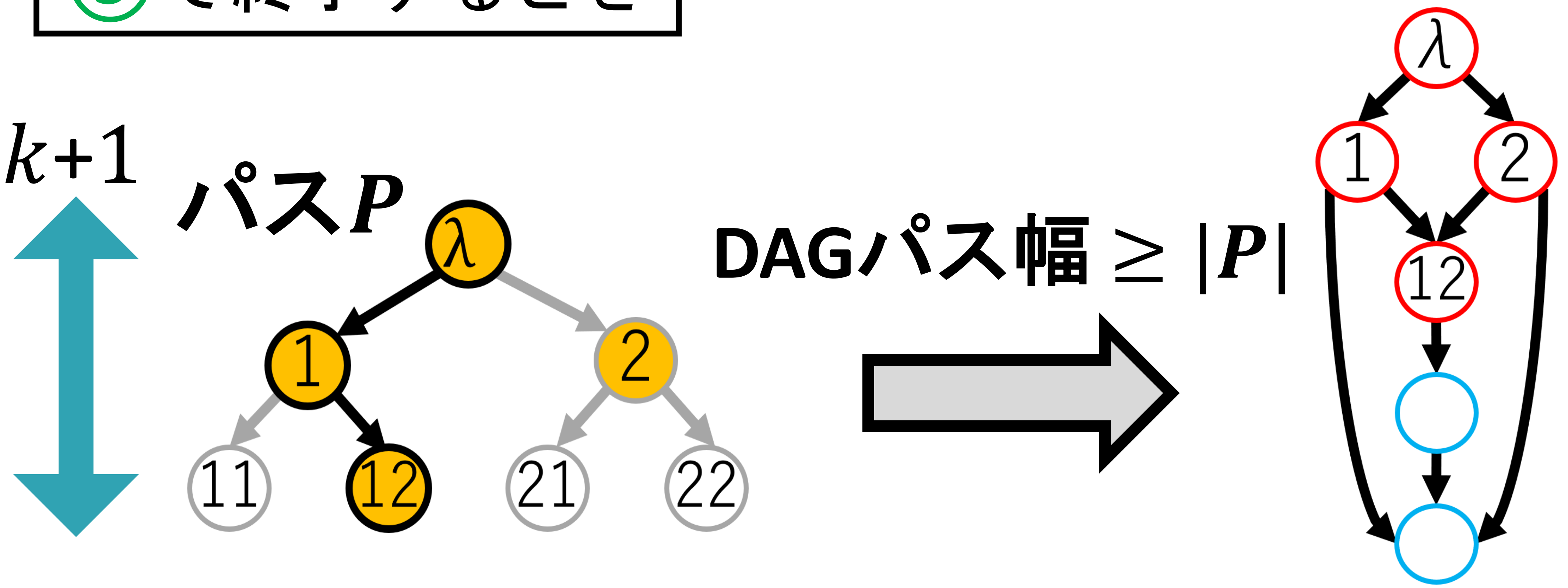
→ M上で根λから葉までの配置済の$token$のみからなる**パスP ($|P|>k+1$)**が存在し，任意のDAGパス分解は必ずあるバッグ$X' \subseteq X_s$ ($|X'| \geq |P| > k+1$)をもつ

1. Gの根に$token\ \lambda$を置く
2. $X_1 \leftarrow GrowTokenTree$. これをDAGパス分解の最初のバッグ
3. until 【Gの全頂点がred】 or 【$|X_i| = |V[M]|$】: ②
   ① if {ある$token\ T$ ($on\ v \in V[G]$)があり, $suc(v)$が全てred, かつTの配置済の子$token$が高々1個}:
      - Tをvから取り除く
      - Tが配置済の子$T \cdot b$をもつ: $T \cdot b \cdot S \leftrightarrow T \cdot S$

   else:
      - return X_i ③

   $X_{i+1} \leftarrow GrowTokenTree$. これを$i$+1番目のバッグ

③で終了するとき



4. まとめ

DAGパス幅に関する種々のアルゴリズムを設計した

①4つのパラメータ化アルゴリズム

有向支配集合問題→ $O(2^w wn)$
最大葉分岐数問題→ $O(2^w wn)$
k-有向点素パス問題→ $O((k+1)^w(w^2+k)\,n)$
k-有向シュタイナー木問題→ $O(2^w(k+w)n)$

n: 頂点数, w: DAGパス幅

③nによらない幅を求めるアルゴリズム

幅が高々$\boldsymbol{O(ld^k)}$のDAGパス分解
→グラフの埋め込みを利用

※l: 根数, d: 最大出次数, k: 入力整数

②DAGパス幅を求める近似アルゴリズム

$\boldsymbol{O(\log^2 n)}$**-近似**
→セパレータを利用
$\boldsymbol{O(\log^{3/2} n)}$**-近似**
→Pebbling gameを利用

④DAG木幅への拡張

DAG木幅→ 有向木への近さを表現
→ DAGパス幅より小

有向支配集合問題→ $O(2^w w^2 n)$

■ 今後の課題

- 幅$O(ld^k)$のdを定数に改善できないか検討
- DAG木幅の利用が適した問題の特徴づけ

コメント用

■コメント