SDNにおける
エンドツーエンドの遅延を考慮したフロー集約法

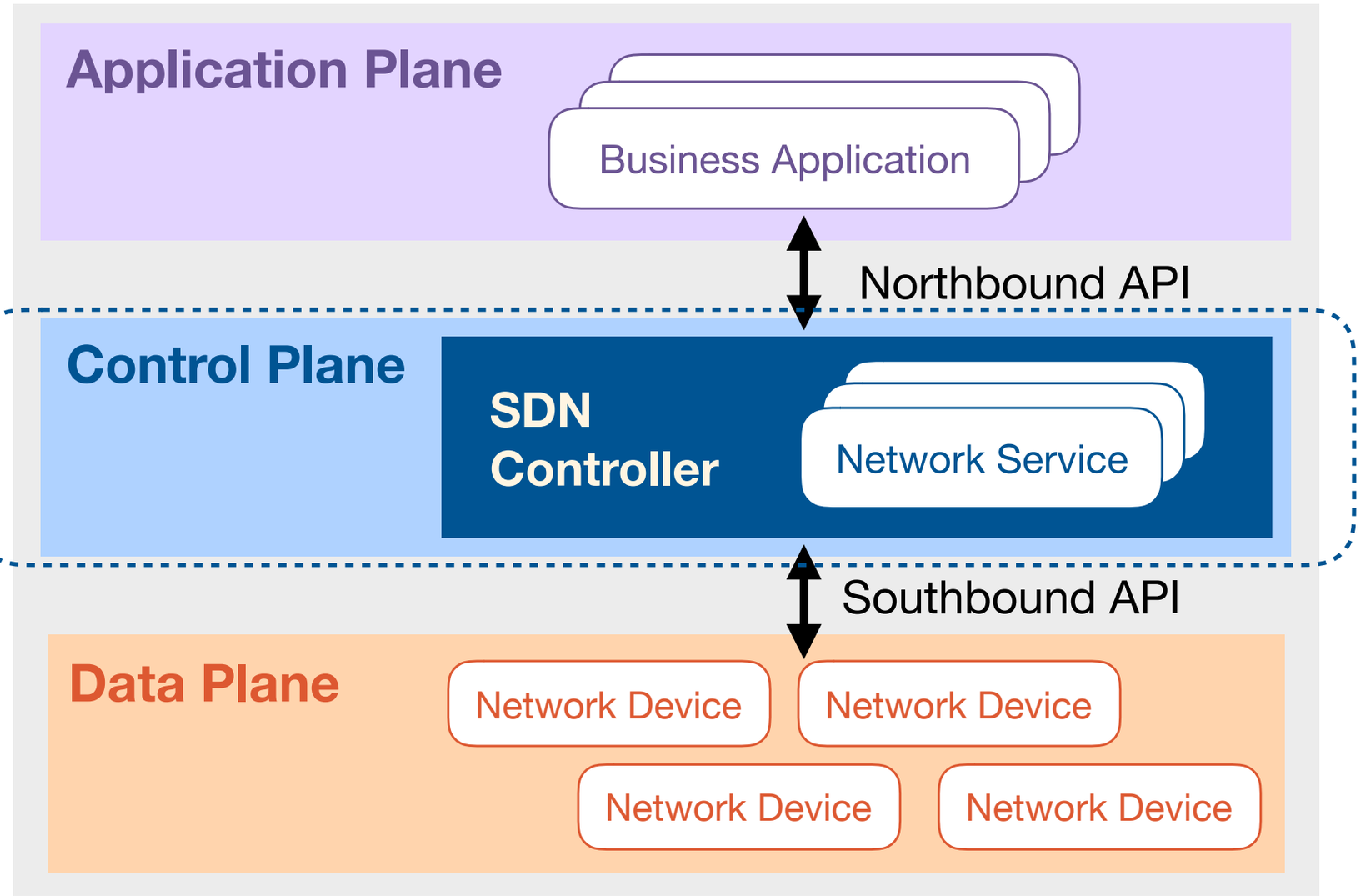
小杉山拓弥† 田辺 和輝† 中山 裕貴‡ 林 經正‡ 山岡 克式†

†東京工業大学

‡株式会社 ボスコ・テスクノロジーズ

研究背景 (1/3)

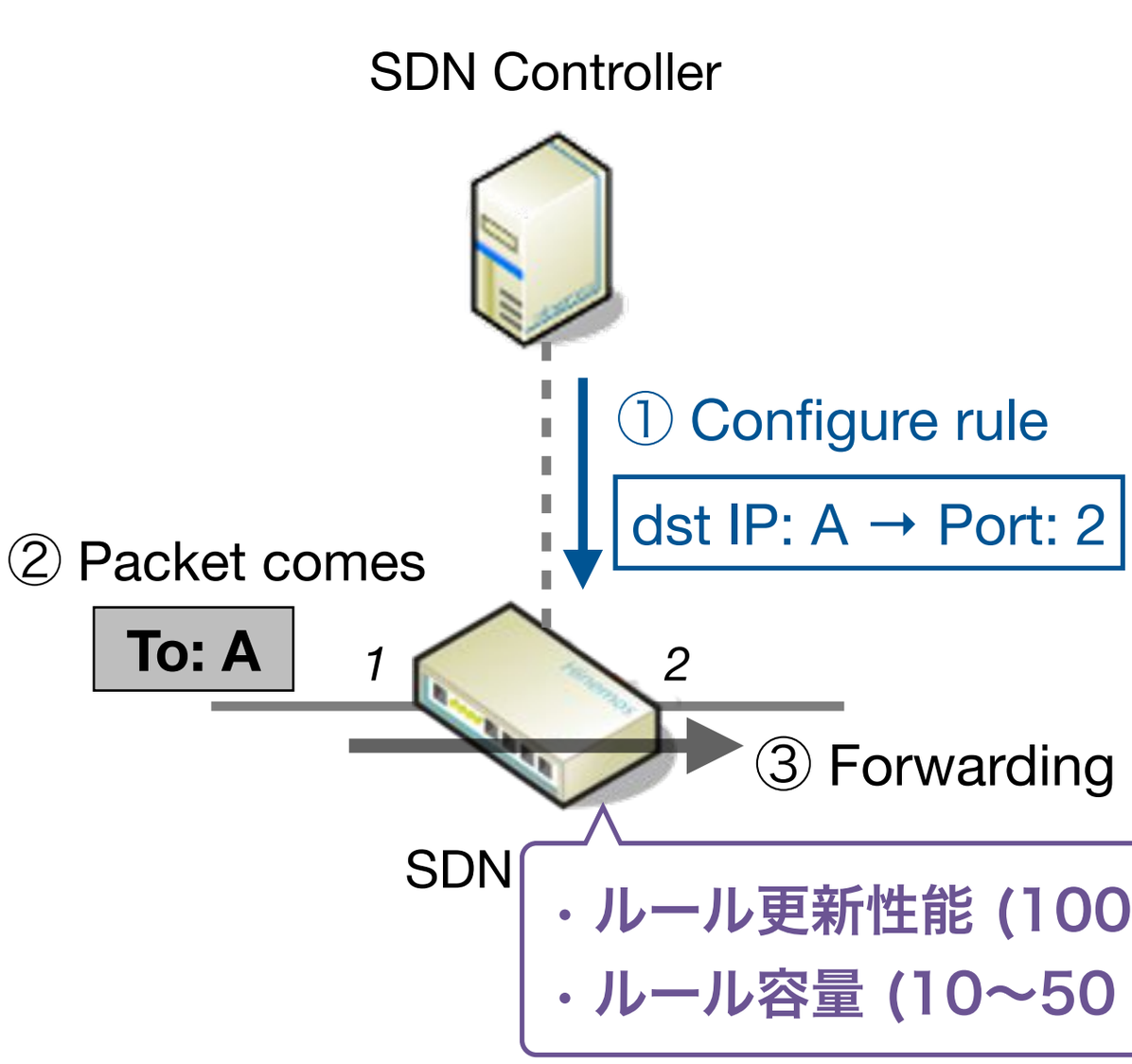
- Software-Defined Networking (SDN)
  - **Cプレーン** と **Dプレーン** を分離した構造



研究背景 (2/3)

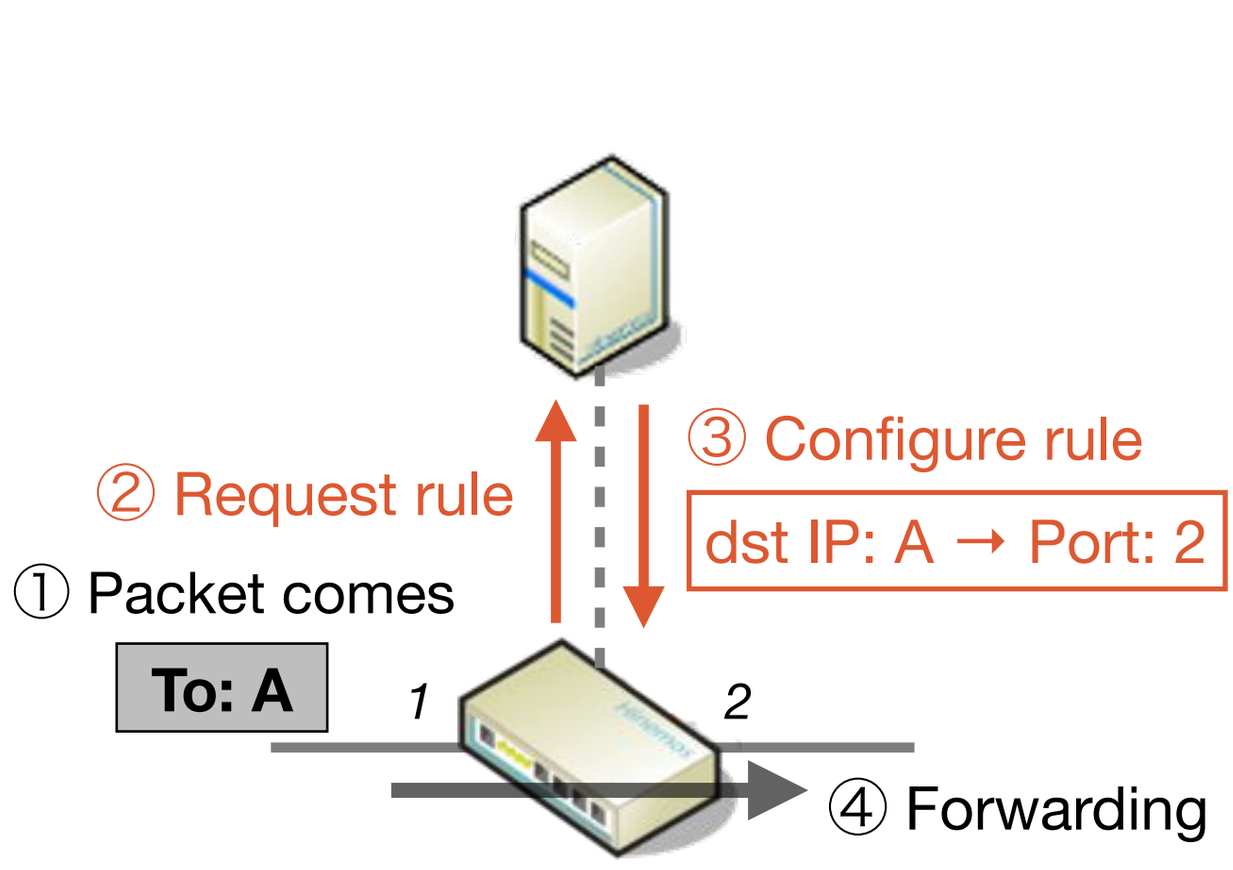
■ SDNにおける経路制御の機構

1. プロアクティブ型 (計画的)



・ルール更新性能 (1000 rules/s)
・ルール容量 (10〜50 K)

2. リアクティブ型 (動的)



多量のフローの管理が困難

研究背景 (3/3)

- SDNにおけるアプリケーションフロー制御
  1. 可能な限りプロアクティブ制御 ⟶ 非現実的
  2. フローを集約しフロー数 (ルール数) を削減
     - 帯域制約を満たすルール集約手法 [1][2]
     - ルール容量を満たすルール集約手法 [3][4]

[1] F. Giroire, J. Moulierac, and T. K. Phan. Optimizing rule placement in software-defined networks for energy-aware routing. In 2014 IEEE Global Communications Conference, pp. 2523–2529, Dec 2014.
[2] Xuan Nam Nguyen, Damien Saucez, Chadi Barakat, and Thierry Turletti. OFFICER: A general optimization framework for OpenFlow rule allocation and endpoint policy enforcement. Proceedings - IEEE INFOCOM, Vol. 26, pp. 478–486, 2015.
[3] Yossi Kanizo, David Hay, and Isaac Keslassy. Palette: Distributing tables in software-defined networks. Proceedings - IEEE INFOCOM, pp. 545–549, 2013.
[4] Nanxi Kang, Zhenming Liu, Jennifer Rexford, and David Walker. Optimizing the ”One Big Switch” Abstraction in Software-Defined Networks. Conext’13, p. 17, 2013.

研究背景 (3/3)

■ SDNにおけるアプリケーションフロー制御

1. 可能な限りプロアクティブ制御 ⟶ 非現実的
2. フローを集約しフロー数 (ルール数) を削減
   - 帯域制約を満たすルール集約手法 [1][2]
   - ルール容量を満たすルール集約手法 [3][4]

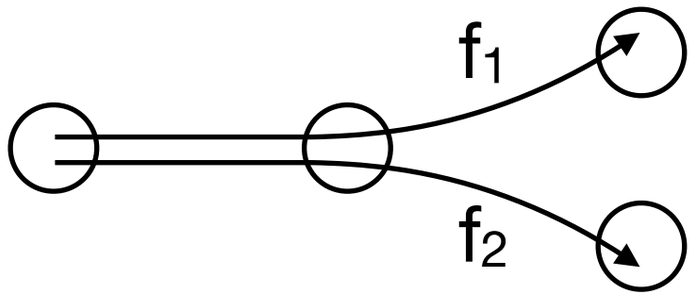
- 許容遅延を考慮して集約する研究は存在しない
  - IoT / M2M 等の許容遅延が小さいフローの需要増加
  - SDNスイッチの制約からフロー数を最小化するべき

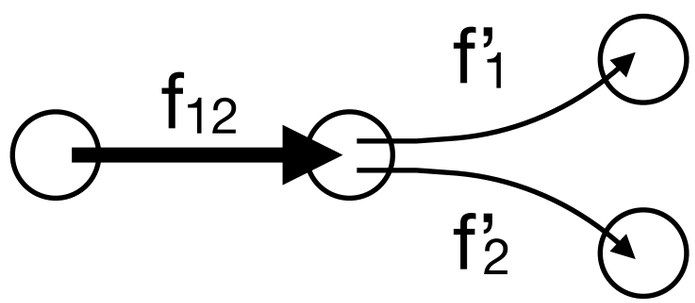
許容遅延を満足しフロー数を最小化する集約法を提案

提案手法

■ フロー集約の考え方

- 同一区間のフローを1つとみなす

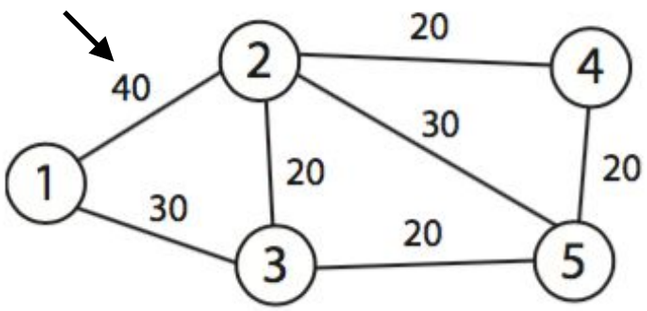




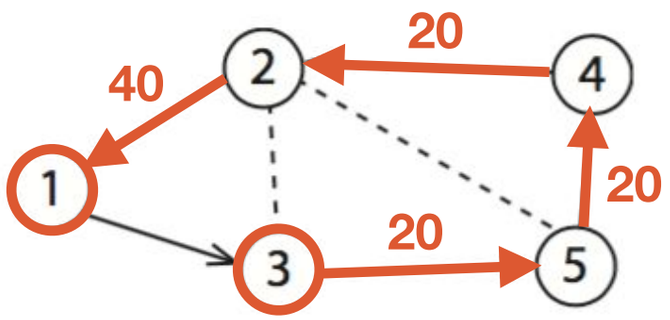
- 集約したフローで経路を構成

遅延を表す
リンクコスト

- 例：フルメッシュにフローが存在（許容遅延＝80）



(a) トポロジ



(b) 許容遅延を満足しない
最大集約 (集約フロー数: 5)

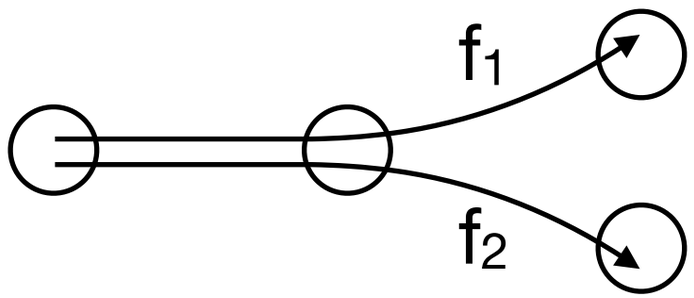
- 3→1のフロー

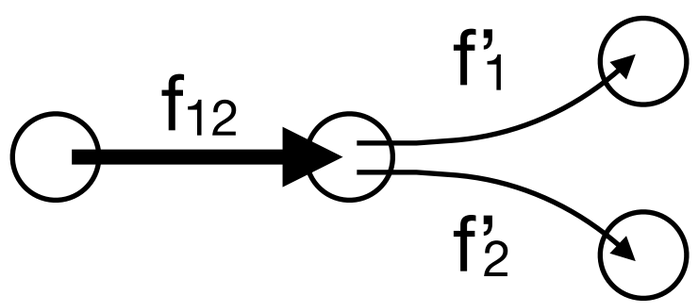
20+20+20+40
= 100 > 80

提案手法

■ フロー集約の考え方

- 同一区間のフローを1つとみなす

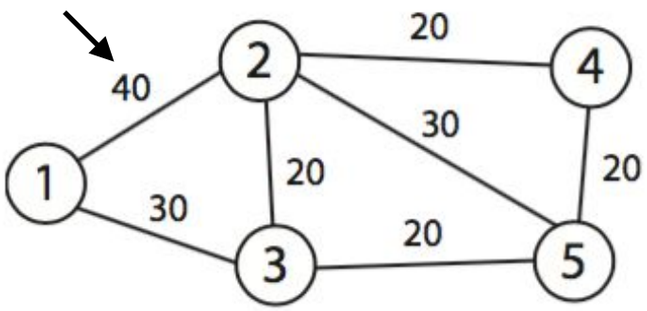




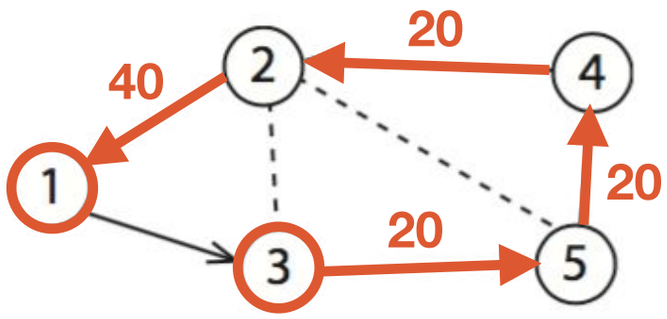
- 集約したフローで経路を構成

遅延を表す
リンクコスト

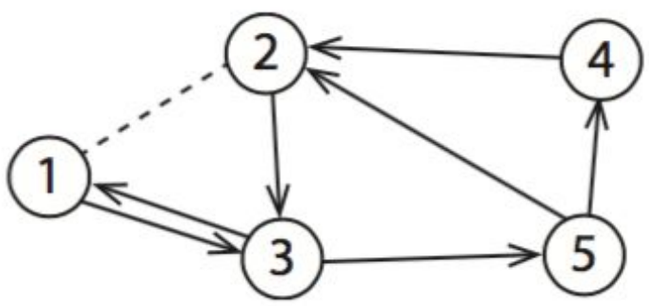
- 例：フルメッシュにフローが存在（許容遅延＝80）



(a) トポロジ



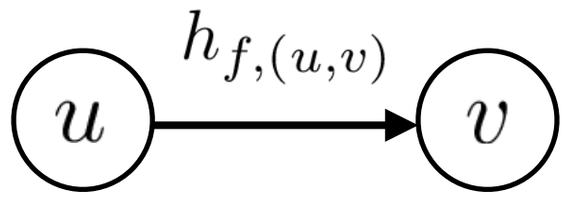
(b) 許容遅延を満足しない
最大集約 (集約フロー数: 5)



(c) 許容遅延を満足する
最大集約 (集約フロー数:7)

問題設定 (1/4)

- SDNを有向グラフ $G = (V, E)$ でモデル化
  - V : SDNスイッチの集合
  - E : リンク $l = (u, v)(u, v \in V, u \neq v)$ の集合
    - リンクコスト $C_{(u,v)} = C_{(v,u)} \in \mathbb{R}^+$
  - F : フロー $f = (f_s, f_d)$ の集合
    - 許容コスト $C_f \in \mathbb{R}^+$
  - フローとリンクの関係



$$h_{f,(u,v)} = \begin{cases} 1 & ((u,v) \text{ is over } f \in F) \\ 0 & (\text{otherwise}) \end{cases} \quad (1)$$

問題設定 (2/4)

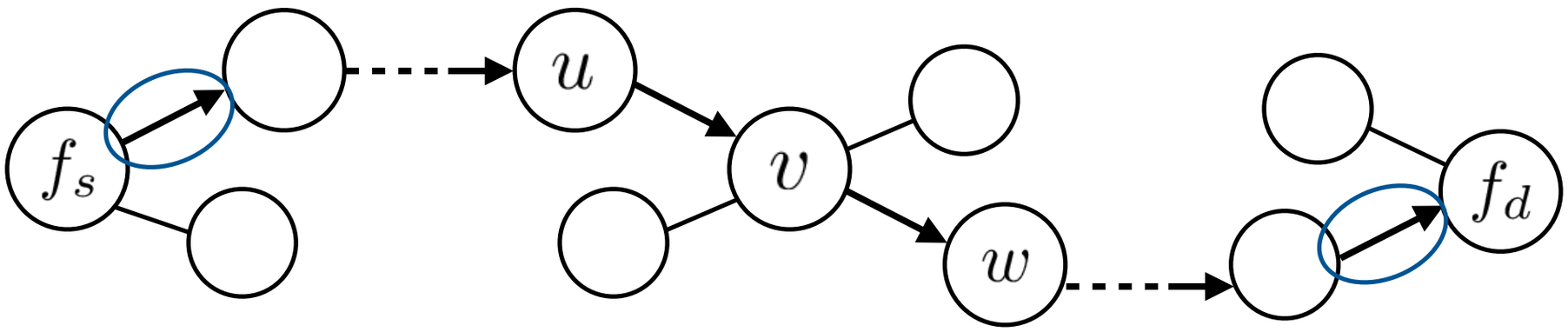
■ フロー到達条件

$$\forall f \in F : \sum_{(f_s,v)\in E} h_{f,(f_s,d)} = \sum_{(u,f_d)\in E} h_{f,(u,f_d)} = 1 \qquad (2)$$

$$\forall f \in F : \sum_{(v,f_s)\in E} h_{f,(v,f_s)} = \sum_{(f_d,u)\in E} h_{f,(f_d,u)} = 0 \qquad (3)$$

$$\forall v \in V \setminus \{f_s, f_d\}, \forall f \in F : \sum_{(u,v)\in E} h_{f,(u,v)} - \sum_{(v,w)\in E} h_{f,(v,w)} = 0 \qquad (4)$$

$$\forall f \in F, \forall v \in V : \sum_{(u,v)\in E} h_{f,(u,v)} = 1 \qquad (5)$$



始点からのフローと終点へのフローは1つのみ

問題設定 (2/4)

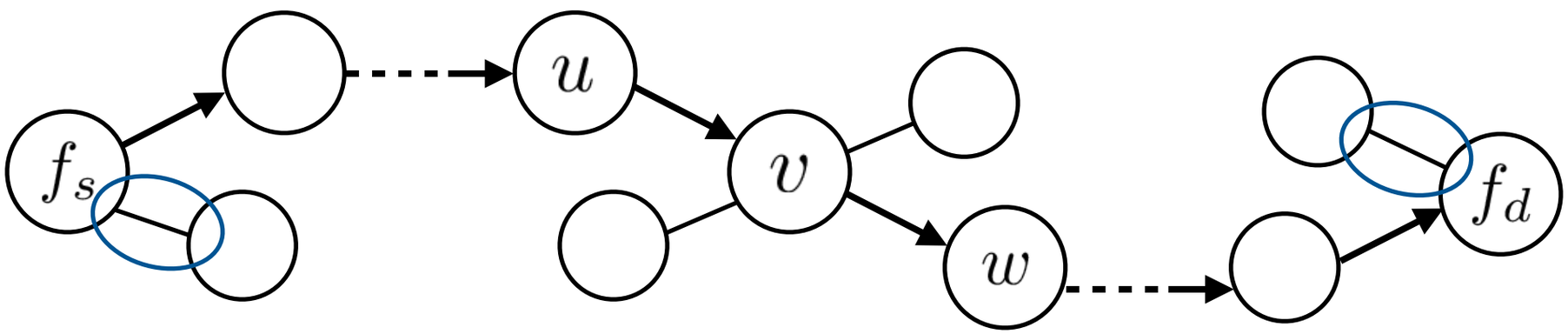
■ フロー到達条件

$$\forall f \in F: \sum_{(f_s,v)\in E} h_{f,(f_s,d)} = \sum_{(u,f_d)\in E} h_{f,(u,f_d)} = 1 \qquad (2)$$

$$\forall f \in F: \sum_{(v,f_s)\in E} h_{f,(v,f_s)} = \sum_{(f_d,u)\in E} h_{f,(f_d,u)} = 0 \qquad (3)$$

$$\forall v \in V \setminus \{f_s, f_d\}, \forall f \in F: \sum_{(u,v)\in E} h_{f,(u,v)} - \sum_{(v,w)\in E} h_{f,(v,w)} = 0 \qquad (4)$$

$$\forall f \in F, \forall v \in V: \sum_{(u,v)\in E} h_{f,(u,v)} = 1 \qquad (5)$$



始点へのフローと終点からのフローは存在しない

問題設定 (2/4)

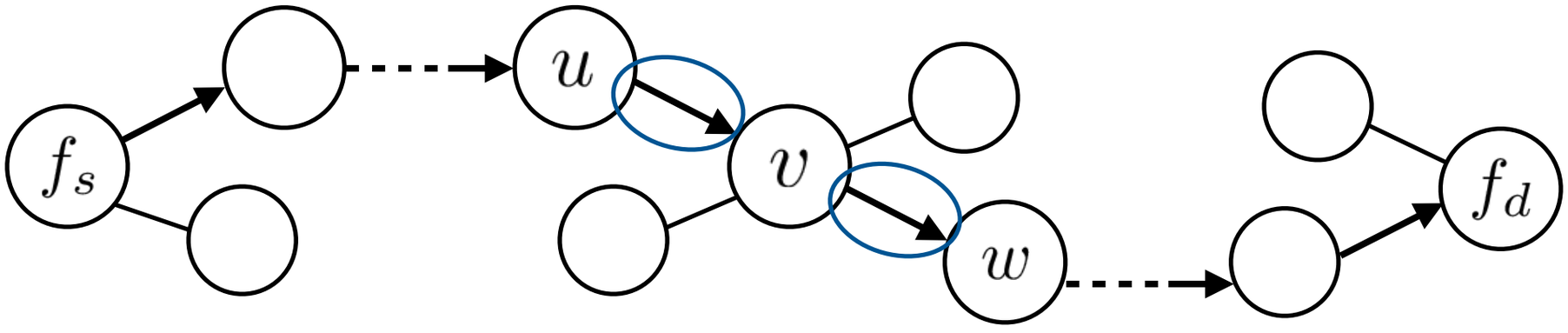
■ フロー到達条件

$$\forall f \in F: \sum_{(f_s,v)\in E} h_{f,(f_s,d)} = \sum_{(u,f_d)\in E} h_{f,(u,f_d)} = 1 \qquad (2)$$

$$\forall f \in F: \sum_{(v,f_s)\in E} h_{f,(v,f_s)} = \sum_{(f_d,u)\in E} h_{f,(f_d,u)} = 0 \qquad (3)$$

$$\forall v \in V \setminus \{f_s, f_d\}, \forall f \in F: \sum_{(u,v)\in E} h_{f,(u,v)} - \sum_{(v,w)\in E} h_{f,(v,w)} = 0 \qquad (4)$$

$$\forall f \in F, \forall v \in V: \sum_{(u,v)\in E} h_{f,(u,v)} = 1 \qquad (5)$$



ノードへ入るフロー数と出るフロー数は等しい

問題設定 (2/4)

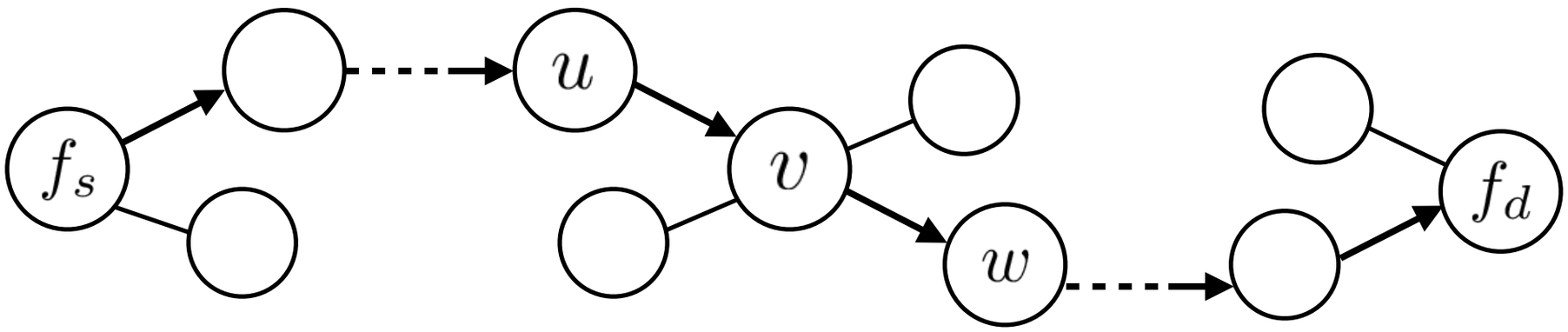
■ フロー到達条件

$$\forall f \in F: \sum_{(f_s,v)\in E} h_{f,(f_s,d)} = \sum_{(u,f_d)\in E} h_{f,(u,f_d)} = 1 \qquad (2)$$

$$\forall f \in F: \sum_{(v,f_s)\in E} h_{f,(v,f_s)} = \sum_{(f_d,u)\in E} h_{f,(f_d,u)} = 0 \qquad (3)$$

$$\forall v \in V \setminus \{f_s, f_d\}, \forall f \in F: \sum_{(u,v)\in E} h_{f,(u,v)} - \sum_{(v,w)\in E} h_{f,(v,w)} = 0 \qquad (4)$$

$$\forall f \in F, \forall v \in V: \sum_{(u,v)\in E} h_{f,(u,v)} = 1 \qquad (5)$$



ノードは1回だけ経由する

問題設定 (3/4)

■ 許容コスト制約

$$\forall f \in F : \sum_{(u,v)\in E} h_{f,(u,v)} C_{(u,v)} \leq C_f \tag{6}$$

■ フロー集約

- 同じリンクを同方向に通過するフローを1つに集約

$$g_{(u,v)} = \begin{cases} 1 & ((u,v) \text{ is over } \exists f \in F) \\ 0 & (\text{otherwise}) \end{cases} \tag{7}$$

$$\forall f \in F, \forall (u,v) \in E : g_{(u,v)} - h_{f,(u,v)} \geq 0 \tag{8}$$

- 目的：フロー数最小化

$$minimize \sum_{(u,v)\in E} g_{(u,v)} \quad \text{s.t.}(2)\text{-}(6)(8) \tag{9}$$

問題設定 (4/4)

■ 式変形

- $\boldsymbol{G} = (g_{(u,v)}, g_{(v,u)}, \cdots)\,(\forall (u,v) \in E)$
- $\boldsymbol{H} = (H_{f_1}, H_{f_2}, \cdots, H_{f_m})\,(f_1, f_2, \cdots, f_m \in F)$
  - $\boldsymbol{H_f} = (h_{f,(u,v)}, h_{f,(v,u)}, \cdots)$
- $\boldsymbol{x} = (\boldsymbol{G}, \boldsymbol{H})$

式 (6)(9) に適用

$$minimize \sum_{(u,v)\in E} \boldsymbol{x}\boldsymbol{z}^T_{(u,v)}$$

$$\text{s.t.} \forall f \in F : \boldsymbol{x}\boldsymbol{y}^T_{f,(u,v)} C_{(u,v)} \leq C_f$$

ナップサック問題に帰着 ⇒ **NP困難**

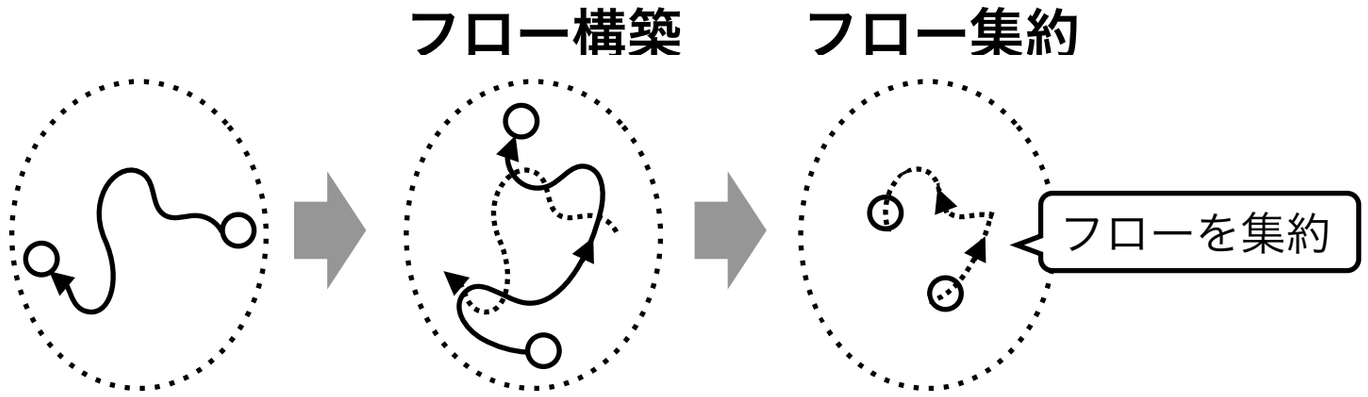
アルゴリズム

■ 全体の流れ

経路変更の自由度

1. フローを(許容コスト – 最小コスト)で昇順ソート ⇒ Fs
2. Fs から1つフローを取り出す ⇒ f
3. [**(1)フロー構築**] 既存ルールを利用して f を構築
4. [**(2)フロー集約**] Fs 中のフローを既存フローに集約
5. Fs が空になるまで 2. に戻る

- 時間計算量 $\mathcal{O}\{|F|((|V|(|E|+|V|))+(|F|(|E|\log|V|)))\}$



(1) フロー構築

■ 最良優先探索で経路構築

- y の評価値 $b_y = \{t_y, c_y\}$
  - t_y：f_s から y に到達したときの既存ルール利用回数
  - c_y：f_s から y に到達したときの合計コスト
- 2ノード間の優先順位付け
  - $comp(u, v) = 1$ ならば v を優先

$$comp(u,v) = \begin{cases} 1 & (t_u < t_v \bigvee (t_u = t_v \bigwedge c_u > c_v)) \\ 0 & (\text{otherwise}) \end{cases}$$

- 最悪の時間計算量
  - $\mathcal{O}(|V|(|E| + |V|))$

(1) フロー構築

■ 最良優先探索で経路構築

・u から v への評価値を求める関数

$$b(u,v,t_{max}) = \begin{cases} \{t_{u+v}, c_{u+v}\} \\ \quad (t_{u+v} < t_{max} \bigwedge c_{u+v} \leq C_f \bigwedge \\ \quad\quad (t_{u+v} > t_v \bigvee \\ \quad\quad\quad t_{u+v} = t_v \bigwedge c_{u+v} < c_v)) \\ b_v \quad (\text{otherwise}) \end{cases}$$

・t_{max} ：ルール使用回数制限値

・t_{u+v} ：$t_u + g_{(u,v)}$

・c_{u+v} ：$c_u + C_{(u,v)}$

フロー構築：例1

- (初期条件) **初期ノードをPQにエンキュー**

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) 隣接ノード z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

(b) $z = f_d$ ならば終了

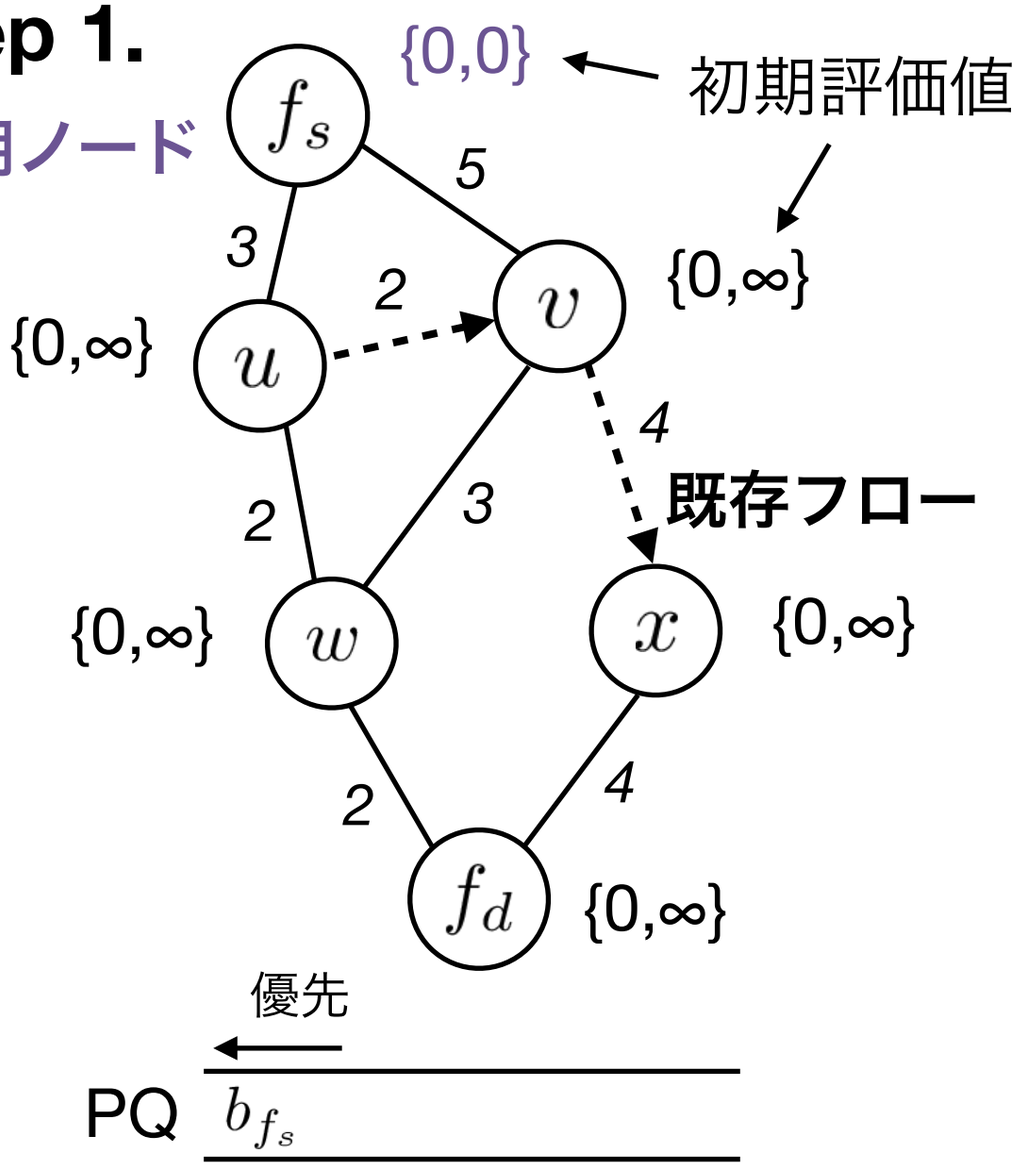
(3) 経由ノードと追加ルールを更新

- y の評価値：$b_y = \{t_y, c_y\}$
  t_y：既存ルール利用回数
  c_y：f_s から y までの合計コスト
- 評価値更新
  優先度が高ければ更新する

(1) $C_f = 13$

Step 1.

初期ノード



- 優先度：(1) t が大きいもの
  (2) t が等しければ c が小さいもの

フロー構築：例1

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) **隣接ノード** z の評価値を計算

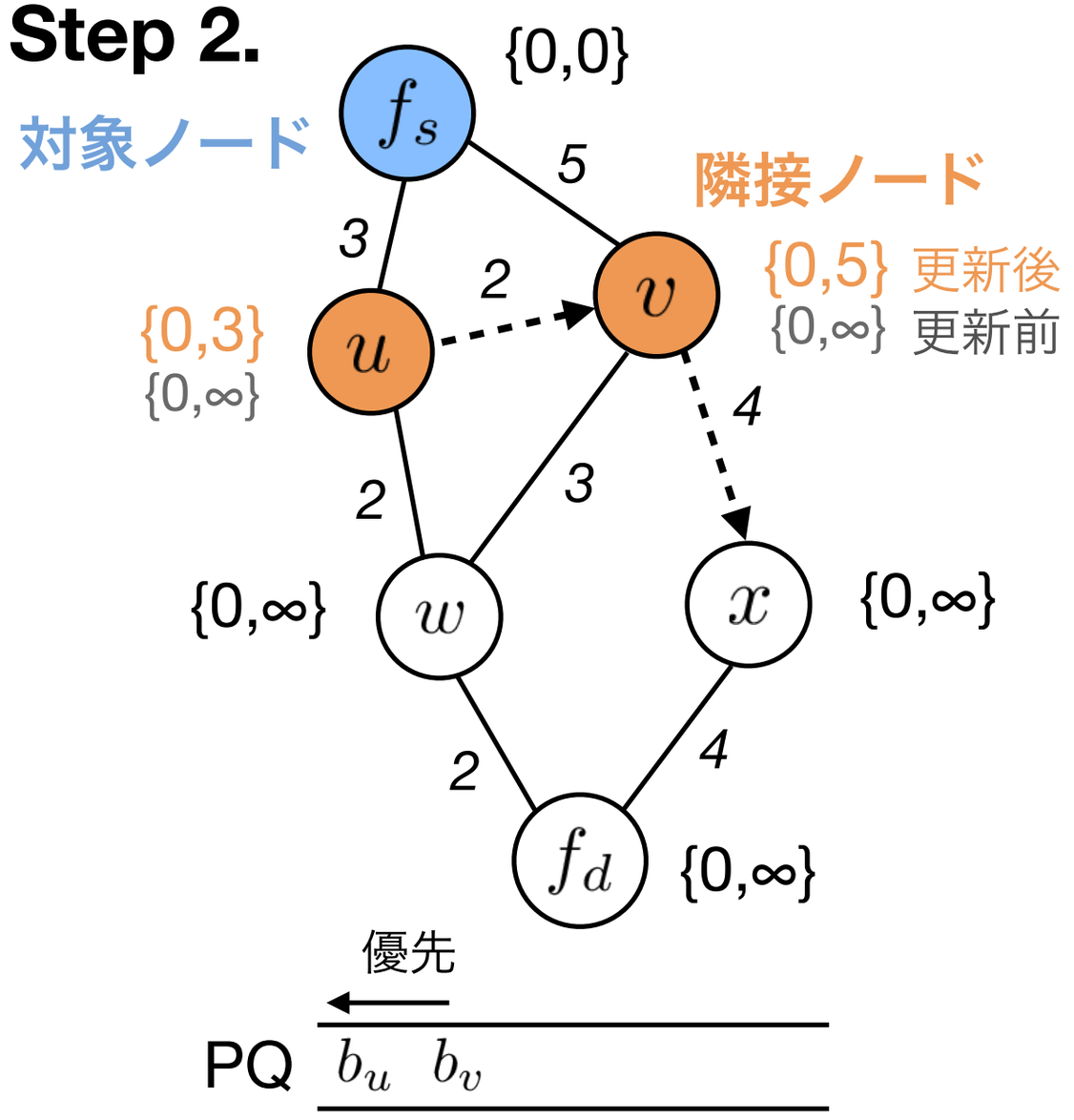
(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

(b) $z = f_d$ ならば終了

(3) 経由ノードと追加ルールを更新

- y の評価値 : $b_y = \{t_y, c_y\}$
  t_y : 既存ルール利用回数
  c_y : f_s から y までの合計コスト
- 評価値更新
  優先度が高ければ更新する

(1) $C_f = 13$



- 優先度 : (1) t が大きいもの
  (2) t が等しければ c が小さいもの

フロー構築：例1

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) **隣接ノード** z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

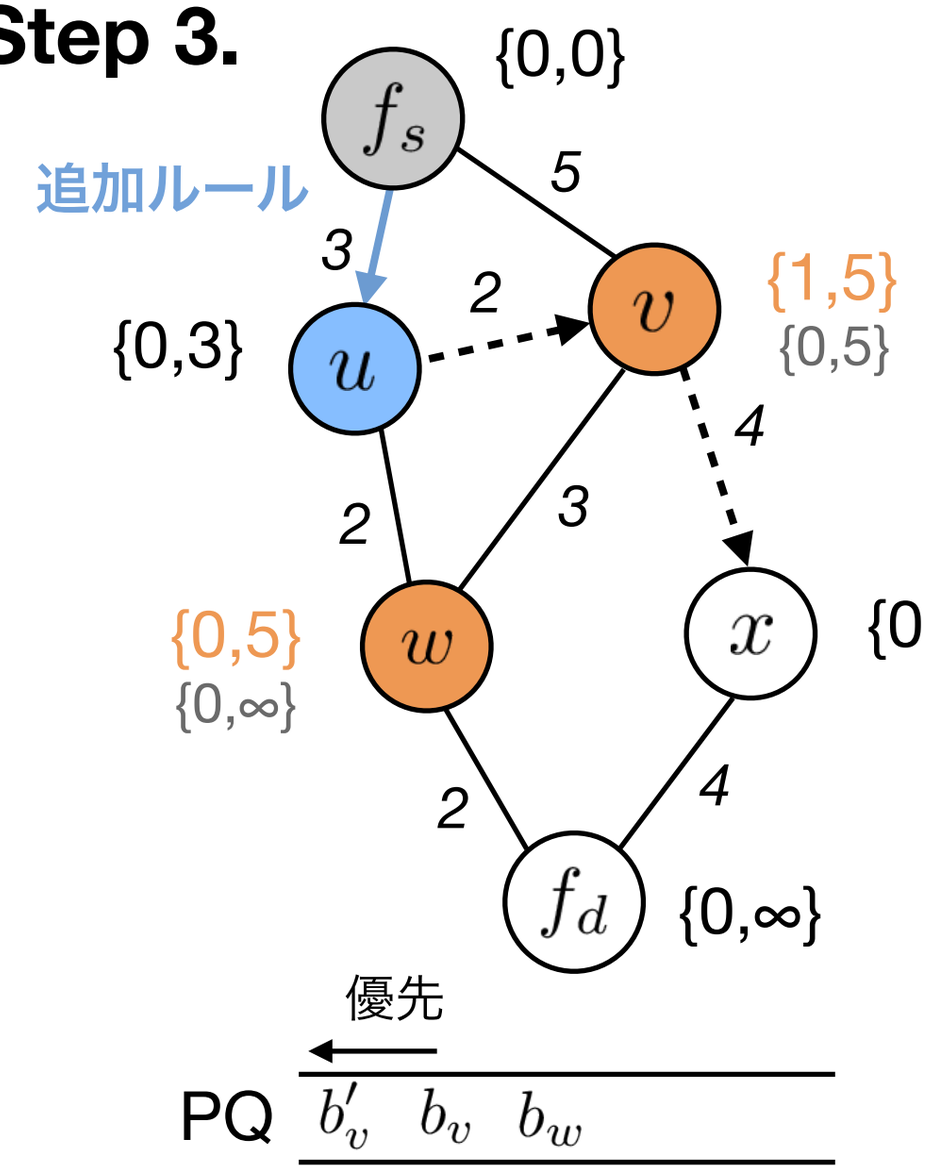
(b) $z = f_d$ ならば終了

(3) **経由ノード**と**追加ルール**を更新

- y の評価値：$b_y = \{t_y, c_y\}$
  t_y：既存ルール利用回数
  c_y：f_s から y までの合計コスト
- 評価値更新
  優先度が高ければ更新する

(1) $C_f = 13$

Step 3.



- 優先度：(1) t が大きいもの
  (2) t が等しければ c が小さいもの

フロー構築：例1

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) **隣接ノード** z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

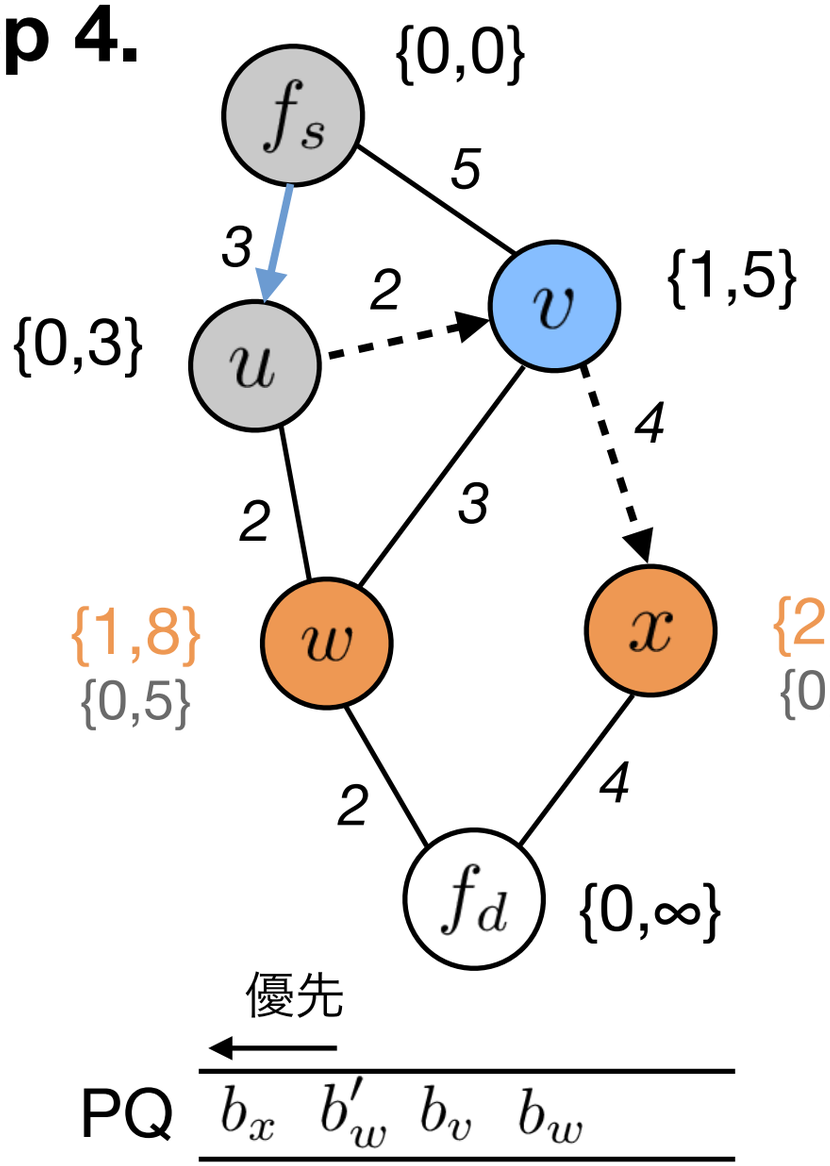
(b) $z = f_d$ ならば終了

(3) **経由ノード**と**追加ルール**を更新

- y の評価値：$b_y = \{t_y, c_y\}$
  t_y：既存ルール利用回数
  c_y：f_s から y までの合計コスト
- 評価値更新
  優先度が高ければ更新する

(1) $C_f = 13$

Step 4.



- 優先度：(1) t が大きいもの
  (2) t が等しければ c が小さいもの

フロー構築：例1

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) **隣接ノード** z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

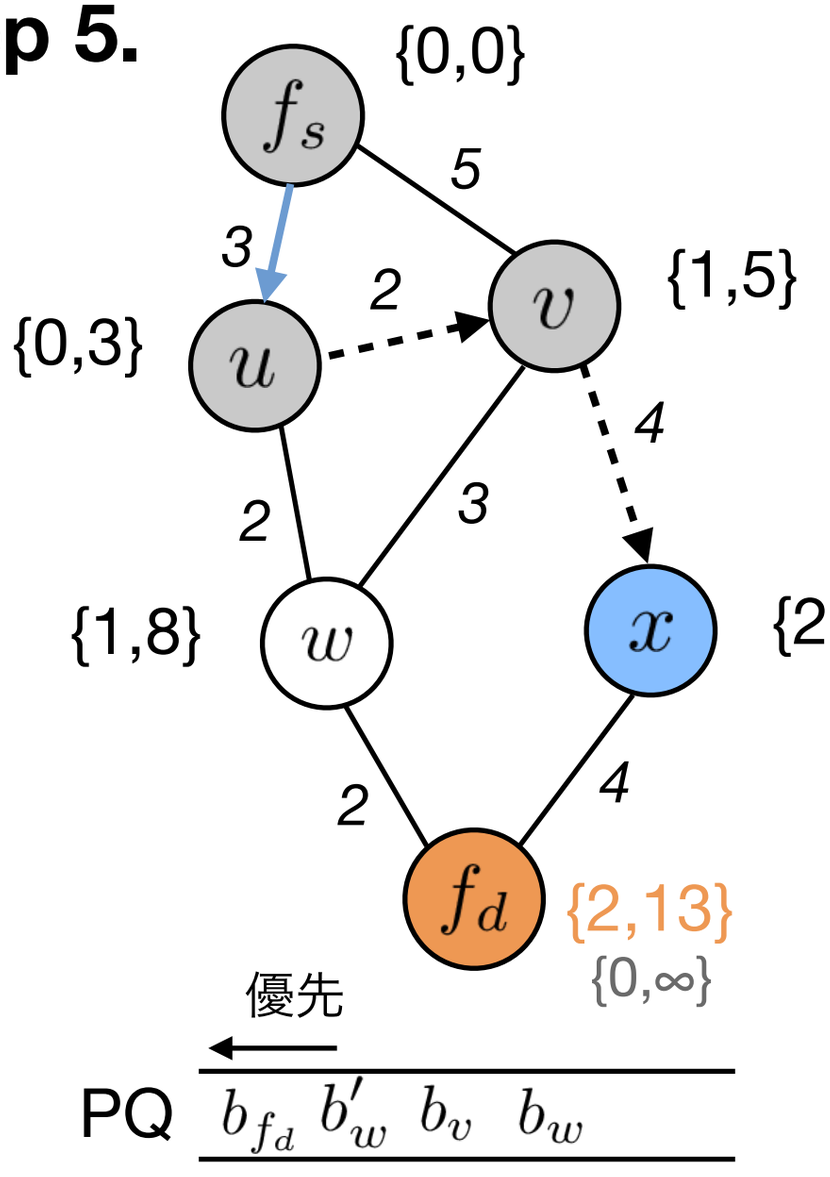
(b) $z = f_d$ ならば終了

(3) **経由ノード**と**追加ルール**を更新

- y の評価値：$b_y = \{t_y, c_y\}$
  t_y：既存ルール利用回数
  c_y：f_s から y までの合計コスト
- 評価値更新
  優先度が高ければ更新する

(1) $C_f = 13$

Step 5.



- 優先度：(1) t が大きいもの
  (2) t が等しければ c が小さいもの

フロー構築：例1

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) 隣接ノード z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

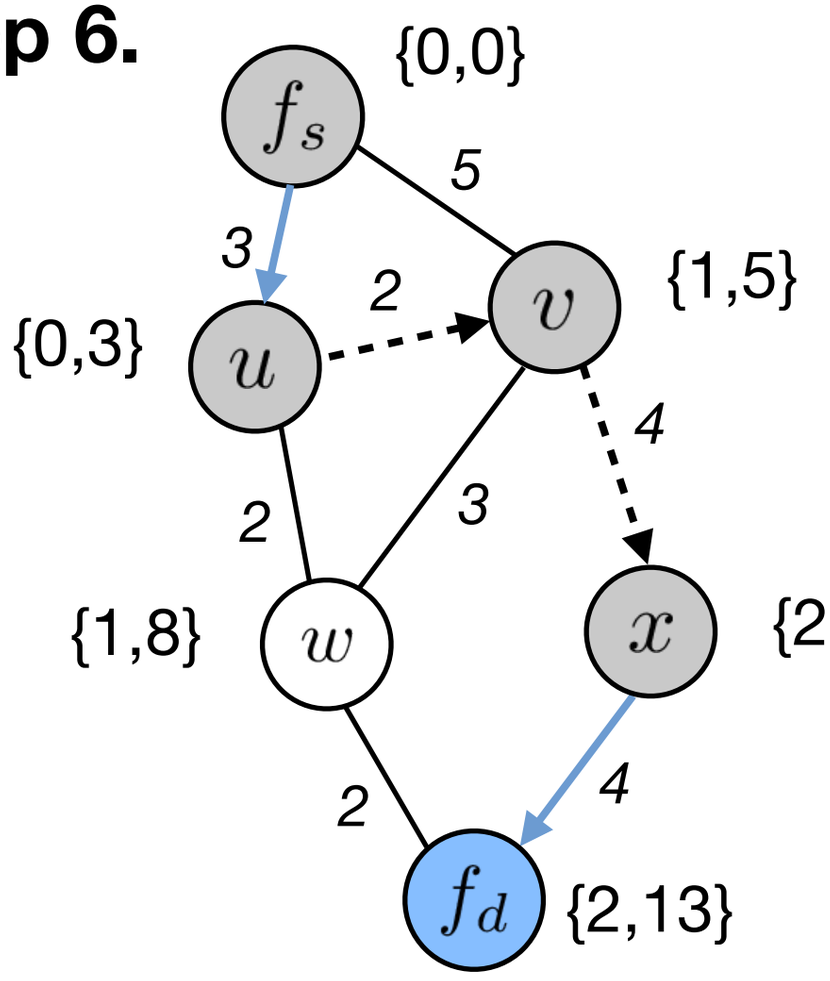
(b) $z = f_d$ ならば終了

(3) 経由ノードと追加ルールを更新

- y の評価値：$b_y = \{t_y, c_y\}$
  t_y：既存ルール利用回数
  c_y：f_s から y までの合計コスト
- 評価値更新
  優先度が高ければ更新する

(1) $C_f = 13$

Step 6.



f_d に到達し C_f を満たしたので終了

追加ルール：$\{\{f_s, u\}, \{x, f_d\}\}$

フロー構築：例2

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) 隣接ノード z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由していなければ評価値を更新して b_z をエンキュー

(b) $z = f_d$ ならば終了

(3) 経由ノードと追加ルールを更新

- y の評価値：$b_y = \{t_y, c_y\}$

  t_y：既存ルール利用回数

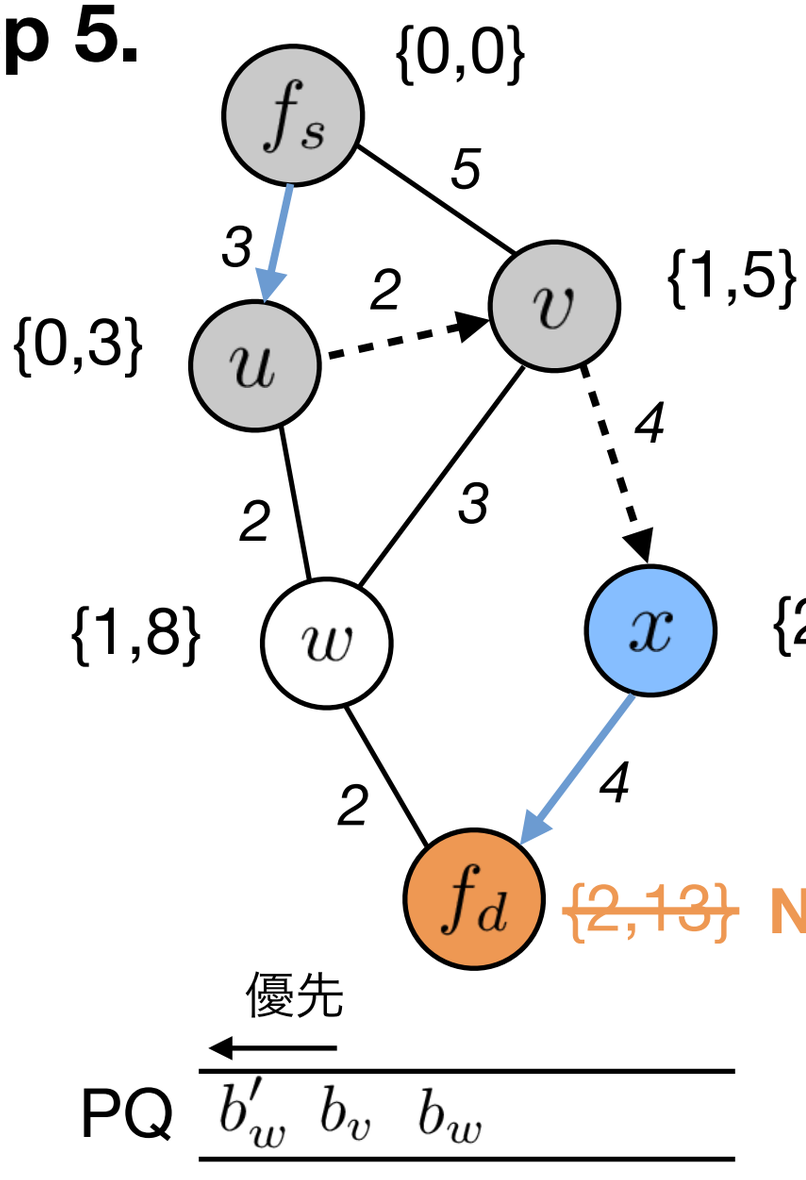
  c_y：f_s から y までの合計コスト

- 評価値更新

  優先度が高ければ更新する

(2) $C_f = 12$

Step 5.



f_d に到達したが C_f を超過

フロー構築：例2

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) **隣接ノード** z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

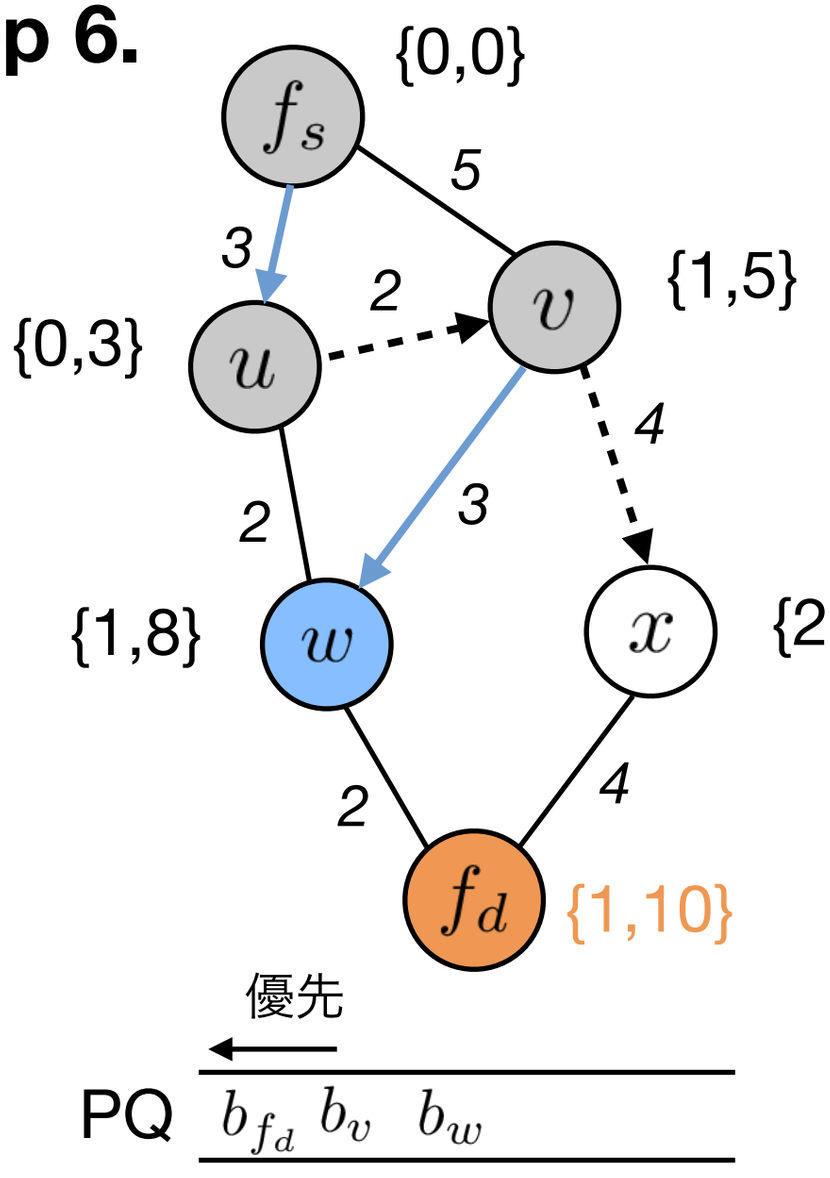
(b) $z = f_d$ ならば終了

(3) **経由ノード**と**追加ルール**を更新

- y の評価値 : $b_y = \{t_y, c_y\}$
  t_y : 既存ルール利用回数
  c_y : f_s から y までの合計コスト
- 評価値更新
  優先度が高ければ更新する

(2) $C_f = 12$

Step 6.



- 優先度 : (1) t が大きいもの
  (2) t が等しければ c が小さいもの

フロー構築：例2

- (初期条件) 初期ノードをPQにエンキュー

Step: PQが空になるまで繰り返す

(1) PQからデキュー ⇒ b_y

(2) 隣接ノード z の評価値を計算

(a) $c_z \leq C_f$ かつ z を経由して
いなければ評価値を更新して b_z を
エンキュー

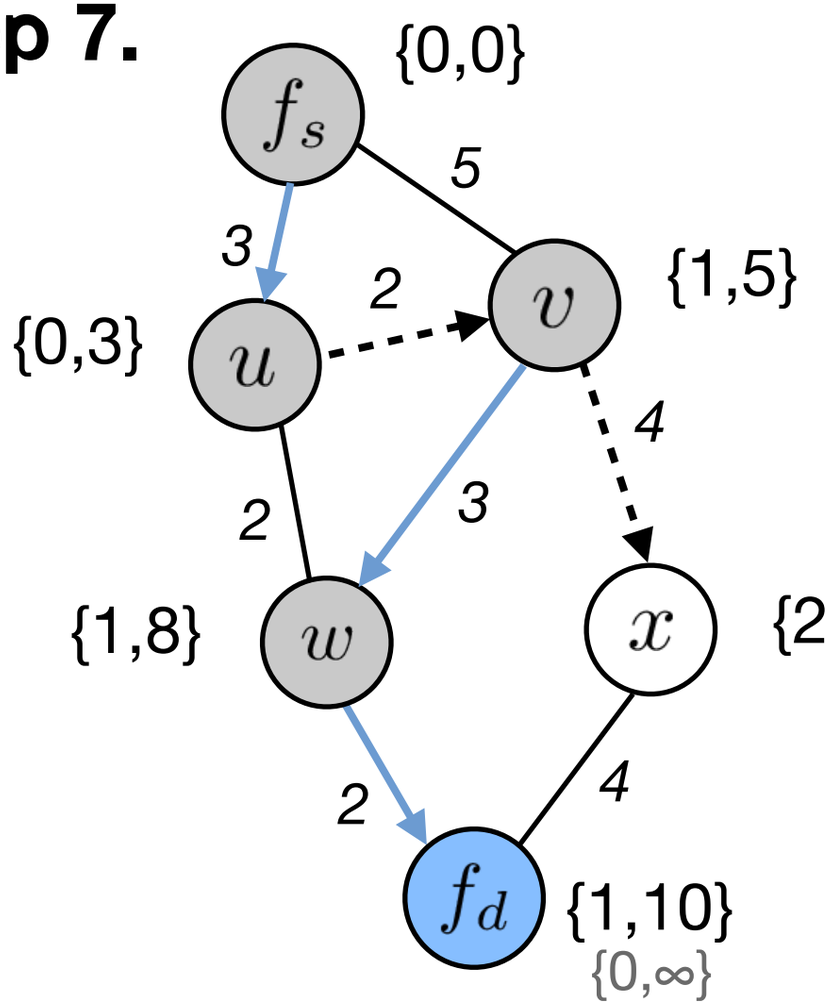
(b) $z = f_d$ ならば終了

(3) 経由ノードと追加ルールを更新

- y の評価値：$b_y = \{t_y, c_y\}$
  - t_y：既存ルール利用回数
  - c_y：f_s から y までの合計コスト
- 評価値更新
  - 優先度が高ければ更新する

(2) $C_f = 12$

Step 7.



f_d に到達し C_f を満たしたので終了

追加ルール：

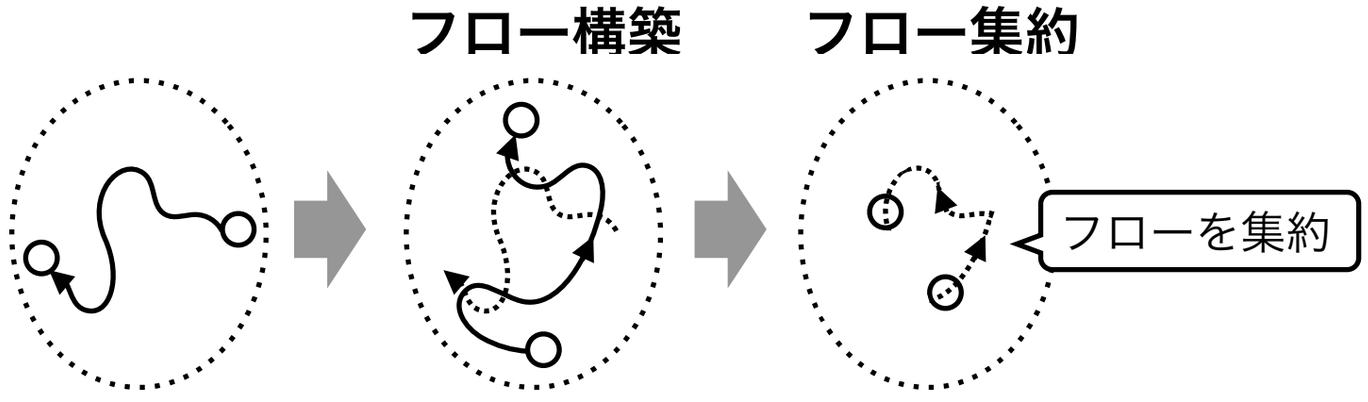
$$\{\{f_s, u\}, \{v, w\}, \{w, f_d\}\}$$

アルゴリズム

■ 全体の流れ

1. フローを (許容コスト – 最小コスト) で昇順ソート ⇒ Fs
2. Fs から1つフローを取り出す ⇒ f
3. [**(1)フロー構築**] 既存ルールを利用して f を構築
4. [**(2)フロー集約**] Fs 中のフローを既存フローに集約
5. Fs が空になるまで 2. に戻る

- 時間計算量 $\mathcal{O}\{|F|((|V|(|E|+|V|))+(|F|(|E|\log|V|)))\}$



(2) フロー集約

- 既存フローと重なっているフローを削除
  - Fs のうち，許容コスト以内で到達できるフローを Fs から削除（ダイクストラ法）
  - 時間計算量
    - $\mathcal{O}(|F|(|E|\log|V|))$

評価 (1)

■ シミュレーションによる評価

- トポロジ
  - モデル：ERモデル / BAモデル
  - 大きさ：|V| = 50, 500
- パラメータ
  - リンク遅延：5〜15のランダム値
  - α：最大の最短遅延に対する許容遅延の設定値の比
- 比較対象
  - 最小コスト / 最小ホップ経路で集約

表：対象トポロジ

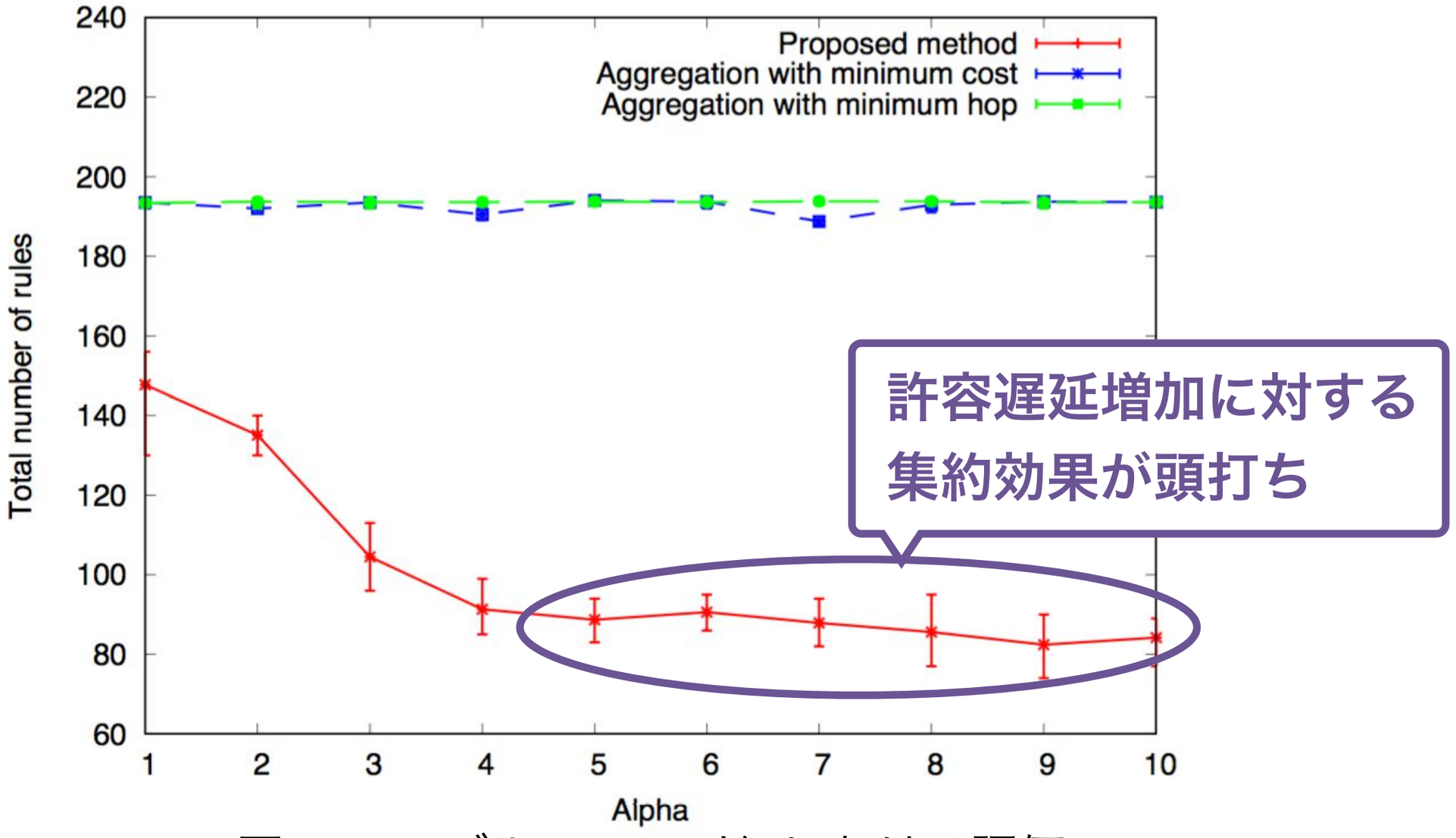
Topology name	$\|V\|$	$\|E\|$	$max(\|F\|)$
Random-small	50	101	$\mathcal{O}(10^3)$
Random-large	500	7472	$\mathcal{O}(10^5)$
Scalefree-small	50	96	$\mathcal{O}(10^3)$
Scalefree-large	500	995	$\mathcal{O}(10^5)$

評価 (2)

■ 許容遅延の変化に対する集約効果

- |F| = 1000 の場合

Topology name	$\|V\|$	$\|E\|$	$max(\|F\|)$
Random-small	50	101	$\mathcal{O}(10^3)$
Random-large	500	7472	$\mathcal{O}(10^5)$
Scalefree-small	50	96	$\mathcal{O}(10^3)$
Scalefree-large	500	995	$\mathcal{O}(10^5)$



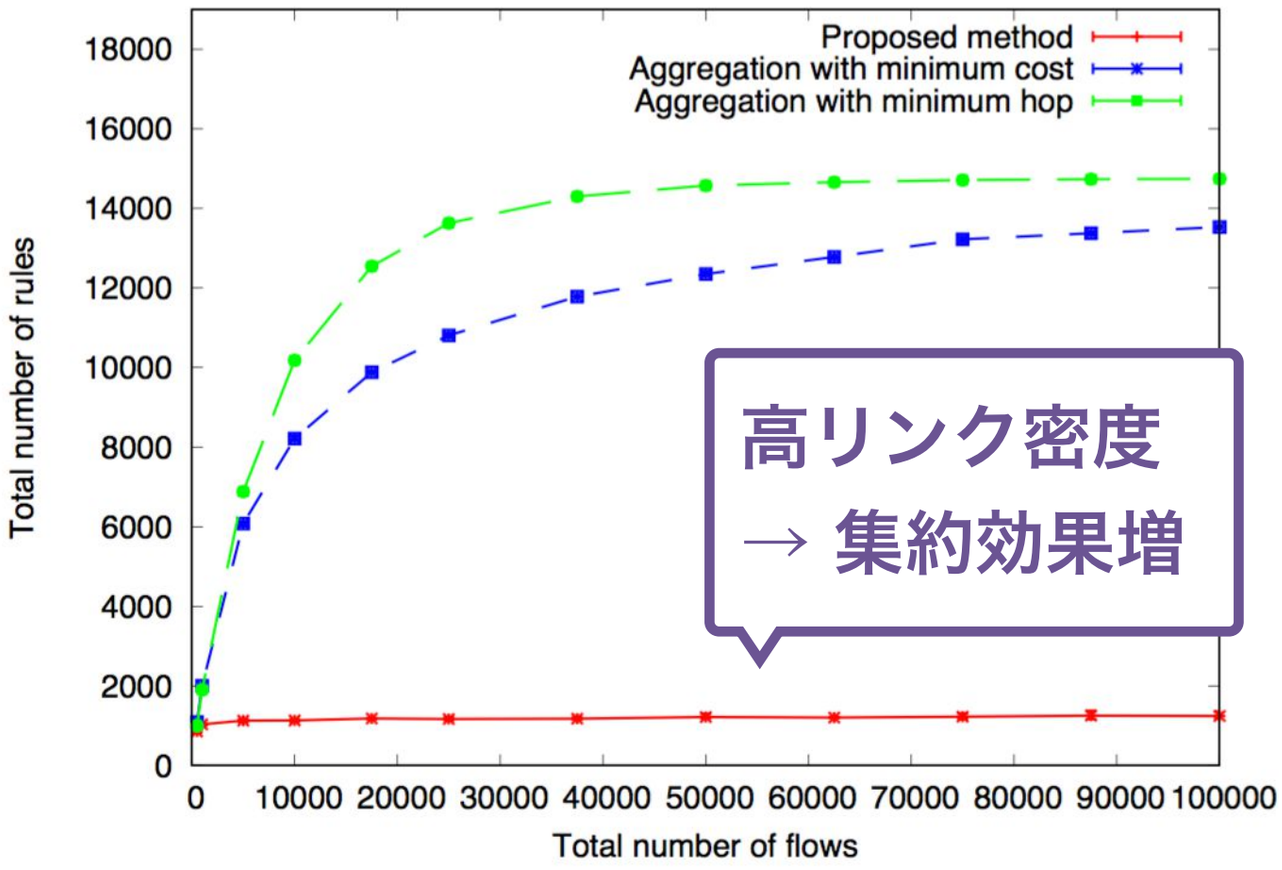
図：ERモデル (50ノード) における評価

評価 (3)

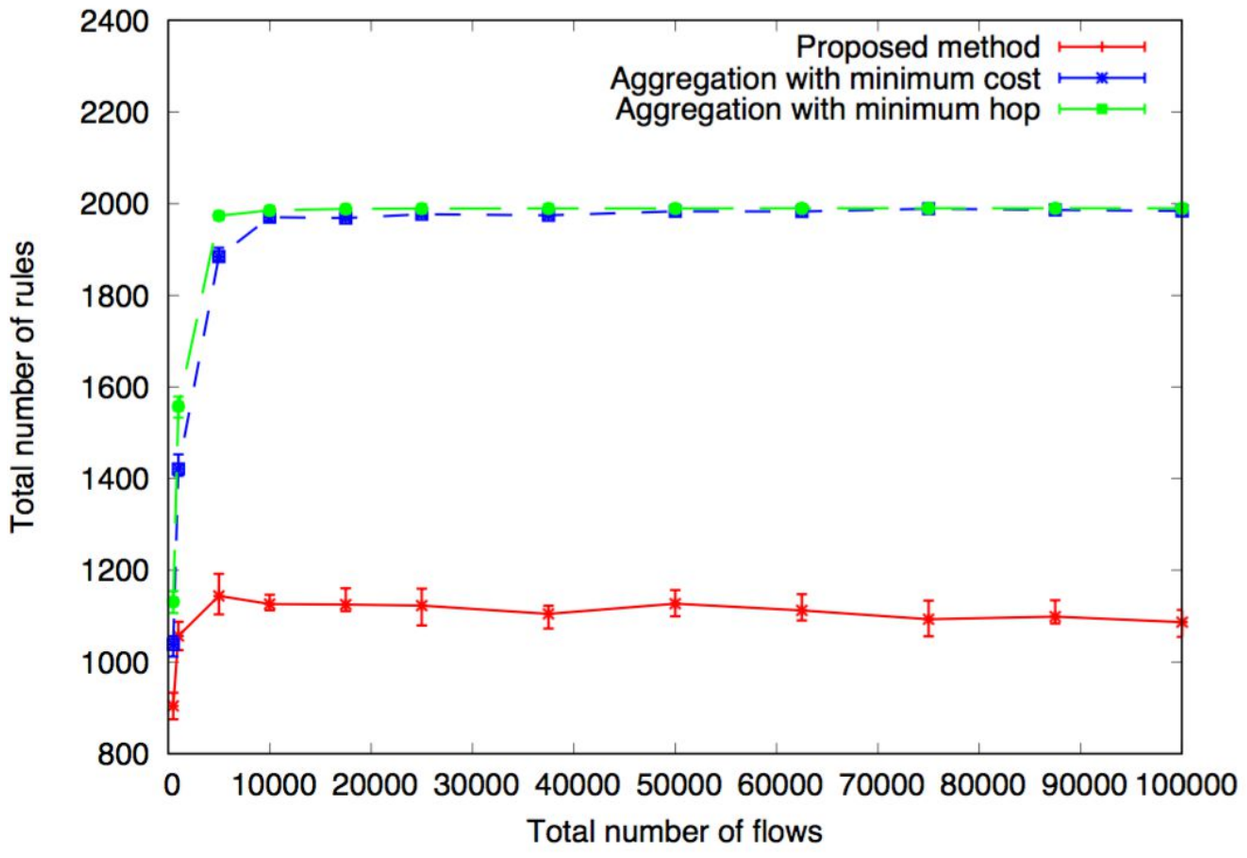
■ フロー数の変化に対する集約効果

- α = 5 の場合

Topology name	$\|V\|$	$\|E\|$	$max(\|F\|)$
Random-small	50	101	$\mathcal{O}(10^3)$
Random-large	500	7472	$\mathcal{O}(10^5)$
Scalefree-small	50	96	$\mathcal{O}(10^3)$
Scalefree-large	500	995	$\mathcal{O}(10^5)$



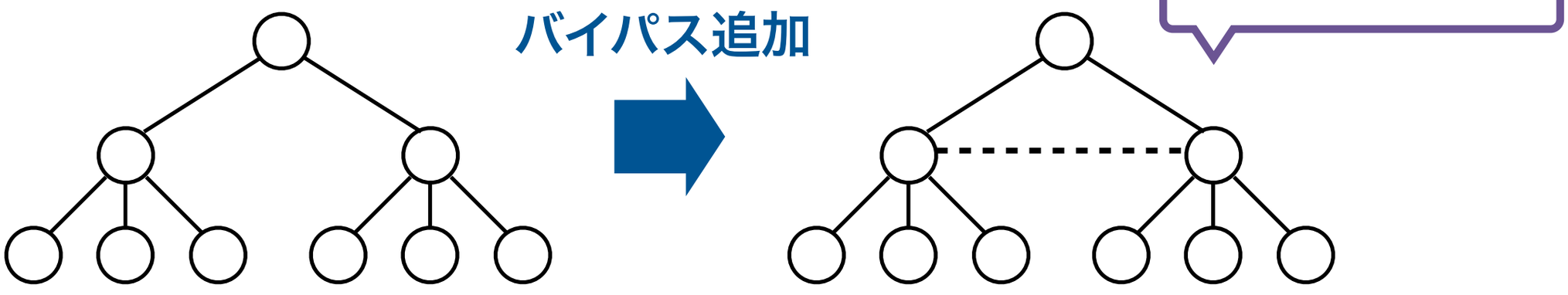
図：ERモデル (500ノード) における評価



図：BAモデル (500ノード) における評価

考察

- 集約効果の要因
  - フローの許容コスト
  - 経路の選択肢数
- 苦手なトポロジ
  - ツリーモデル
    - 経路が一意に決定



まとめ

- エンドツーエンドのフローの許容遅延を満足しフロー数を最小化する集約法を提案
- 複数トポロジでシミュレーション評価
  - 単純な手法より高い集約効果
  - 経路変更の自由度が影響
- 今後の課題
  - 現実的なモデルへの修正
  - 苦手なトポロジのバイパス追加手法検討