

アルゴリズムとデータ構造⑩

～ 最大流問題 ～

鹿島久嗣

最大流問題

ネットワーク：

辺に容量をもった、入口と出口をもつ有向グラフ

■ グラフ $G = (V, E)$ ：頂点を辺（＝枝）でつないだもの

– V ：頂点集合（有限集合）

– E ：辺の集合（ V 上の2項関係； $E \subseteq V \times V$ ）
直積集合

• 辺 $e = (v, w)$ に向きがある場合が有向グラフ

■ ネットワーク：

– 特別な頂点 v_s （入口）と v_t （出口）をもつ有向グラフ

– 各辺 $e = (v, w)$ が容量 $c(e) \geq 0$ をもつ

※ 以下、簡単のため始点からは出る方向の辺のみ、
終点には入る方向の辺のみがあるとする（これは本質的でない）

ネットワークのフロー：

ネットワークの入り口から出口までモノを流す

- ネットワーク上で入口から出口までモノを流すことを考える
- フロー $f: E \rightarrow \mathbb{R}_+$
 - $f(e) \geq 0$ は辺 $e \in E$ の上を通す物量のようなもの
 - ただし、 $f(e) \leq c(e)$ ：辺の容量より多くは通せない
- (出入り口以外の) 各頂点 v において以下の関係が成立

$$\sum_{e \in V^+(v)} f(e) = \sum_{e \in V^-(v)} f(e)$$

出入りのバランスが
取れている

- $V^+(v)$ は v に入る辺； $V^-(v)$ は v から出る辺

ネットワークのフロー :

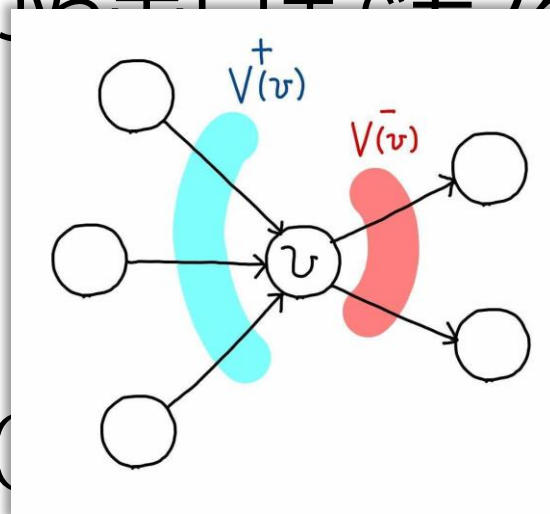
ネットワークの入り口から出口までモノを流す

- ネットワーク上で入口から出口までモノを流すことを考える

- フロー $f: E \rightarrow \mathbb{R}_+$

- $f(e) \geq 0$ は辺 $e \in E$ を通るモノの量

- ただし、 $f(e) \leq c(e)$ (容量) より多くは通せない



- (出入り口以外の) 各頂点 v において以下の関係が成立

$$\sum_{e \in V^+(v)} f(e) = \sum_{e \in V^-(v)} f(e)$$

出入りのバランスが
取れている

- $V^+(v)$ は v に入る辺 ; $V^-(v)$ は v から出る辺

ネットワークの最大流問題： 容量制約を満たしながら、できるだけモノを流す

■最大流問題：

- 入口から出る量（＝出口に入る量）を最大化する

$$\max_f \sum_{e \in V^-(v_s)} f(v_s) \left(= \max_f \sum_{e \in V^+(v_t)} f(v_t) \right)$$

$$\text{s. t.} \quad 0 \leq f(e) \leq c(e) \quad \forall e \in E$$

- 問題の解として決定すべきはフロー f （各辺に通す量）

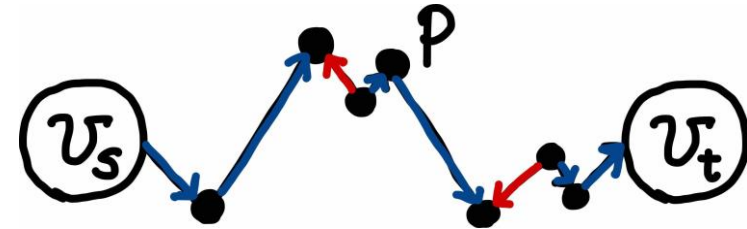
最大流問題のアルゴリズム： フォード・ファルカーソン

- フォード・ファルカーソンの基本的な考え方：
現在の解（フロー）に新たなフローを逐次的に足していく
 - 現在のフロー f があるとする
 - ある条件（後述）を満たす v_s から v_t へのパス p をみつける
 - このパス p に沿って追加のフロー Δf を流す
$$f(e) \leftarrow f(e) + \Delta f(e) \text{ for } \forall e \in p$$
 - 条件（後述）を満たすパスがなくなるまで、
以上を繰り返す



フォード・ファルカーソンにおける解の更新： フローを追加できる「余裕」のあるパスを見つける

- v_s から v_t への（辺の向きを無視した）パス p を考える
- パスに含まれる各辺 $e = (v, w) \in p$ に対して 2 つの場合：
 1. **正順**：パスの向きと、辺の向きが一致している場合
 2. **逆順**：一致していない場合



- 各辺 e について「余裕」 $g(e)$ を考える
 - **正順**の場合： $g(e) = c(e) - f(e)$ ：新たに流せる余裕
 - **逆順**の場合： $g(e) = f(e)$ ：逆向きに流せる
(減らせる) 余裕がある

フォード・ファルカーソンにおける解の更新：
フローを追加できる「余裕」のあるパスを見つける

- 「パスの余裕」をパス上の辺の余裕の最小値で定義する：

$$g(p) = \min_{e \in p} g(e)$$

- $g(p)$ が正であれば、このパスに沿ってさらに $g(p)$ の物流を新たに流せるはず

- 正順の辺については物流を増やす：

$$f(e) \leftarrow f(e) + g(p)$$

- 逆順の辺については物流を減らす（逆向きに流す）：

$$f(e) \leftarrow f(e) - g(p)$$

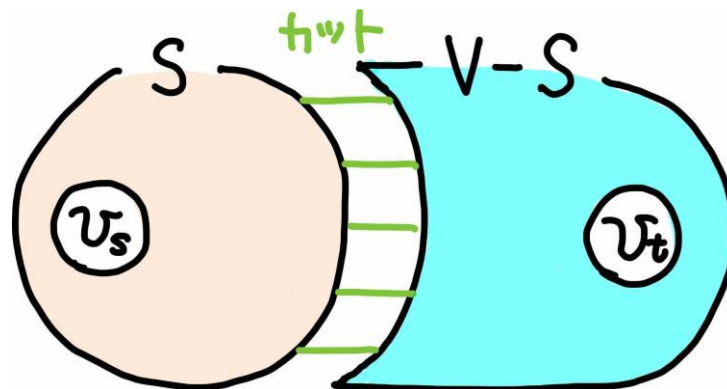
余裕のあるパスの発見： たとえば、幅優先探索

- $g(p)$ が正であるようなパス p を見つけるには？
 - ーフォード・ファルカーソンでは p のを見つけ方は決められていない
- たとえば、幅優先探索で正の余裕をもつパスの中で最も短いパスを用いる（＝エドモンズ・カープ法）
 - ー幅優先探索で $g(e)$ が正である（余裕のある）辺をつなげていく
 - ー以前見た、グラフの頂点列挙と同じ方式

カット：

最大フローと最小カットは等しい

- カット： v_s を含む頂点集合 $S \subseteq V$ から出て、 $V - S$ に含まれる頂点のいずれかに入る辺の集合
- 最大フロー最小カット定理：
カットに含まれる辺の容量の和の最小値 = 最大流量
- 気持ち：どんなカットをとっても、必ずフローの流量が、 S から $V - S$ に流れているはずなので、実際に流せるのはその中で最小の量のはず



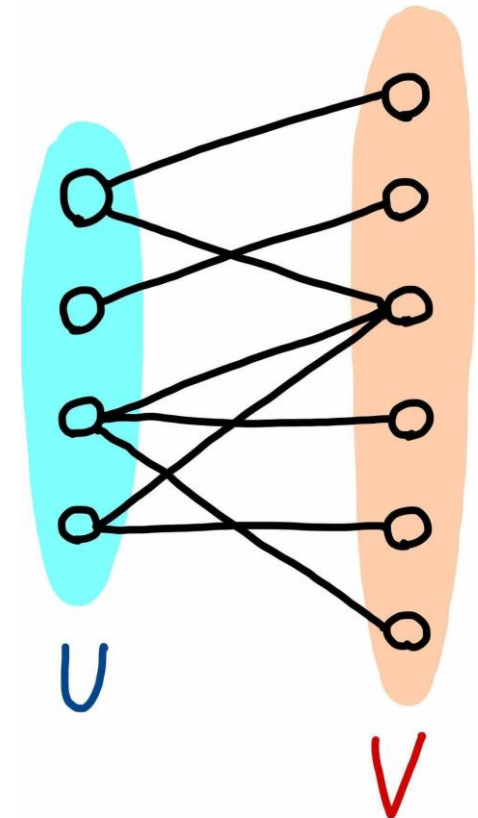
マッチング

2部グラフ :

2つの集合の関係を表すグラフ

■ 2部グラフ

- 頂点集合が2つの集合 U と V に分かれている
- 枝は2つの集合間にしか存在しない



2部グラフのマッチング：

2グループ間で成立するペアの数を最大化する

■ マッチング問題：

– 2部グラフにおいて、以下の条件を満たす辺の集合を選ぶ

- すべての頂点は、たかだか 1 つの辺にしか含まれない

– もっとも大きい辺の集合を見つける

■ 例：マッチングアプリ的な何か

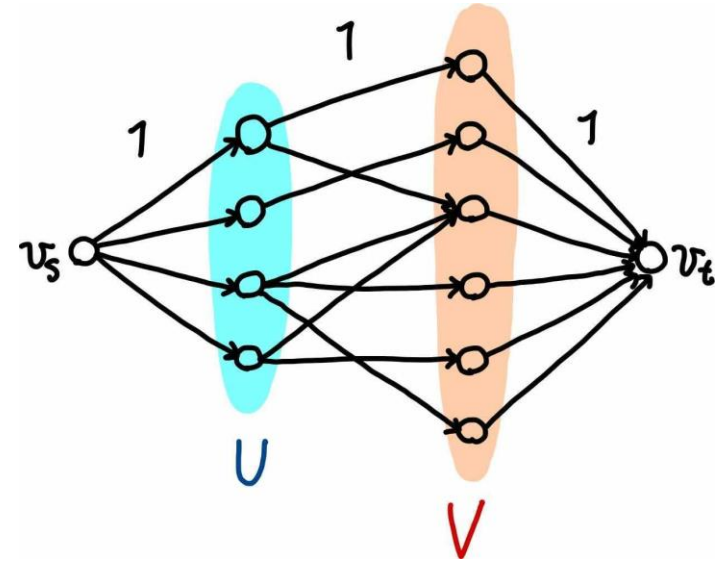
– 2つのグループ間で、各人が（互いに）パートナーを組んでいいと思う人の間に辺があるとする

– 成立するペアの数を最大化したい

マッチング問題のアルゴリズム： 最大流に帰着できる

■ 最大流問題に帰着可能

- v_s から U の各頂点に容量1の辺
- U と V の間の辺に容量1
- U から v_t の各頂点に容量1の辺



■ フォード・ファルカーソンを適用すればマッチングが得られる

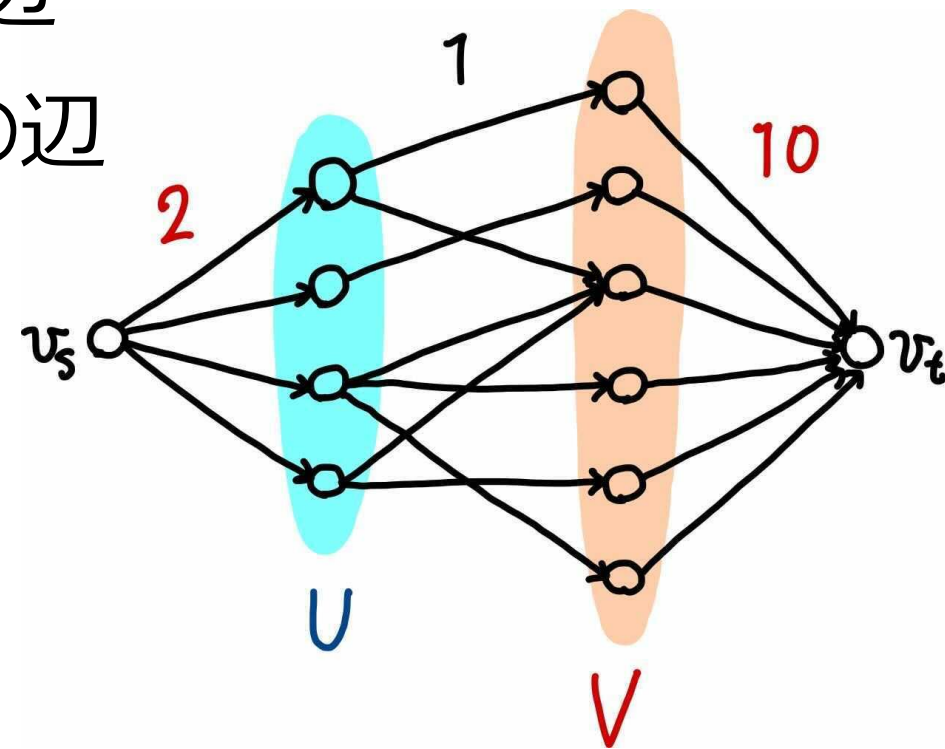
- フォード・ファルカーソンでは、
すべての辺の容量が整数であれば、その解も整数
- この場合、 U と V の間の辺を流れる量は0か1

割り当て問題： マッチング問題の一般化

- マッチングを多対多の対応に一般化
- 学生とゼミのマッチング：
 - 学生は2つのゼミに所属
 - 各ゼミは最大10名まで受入れ可能

割り当て問題のアルゴリズム： やはり最大流に帰着できる

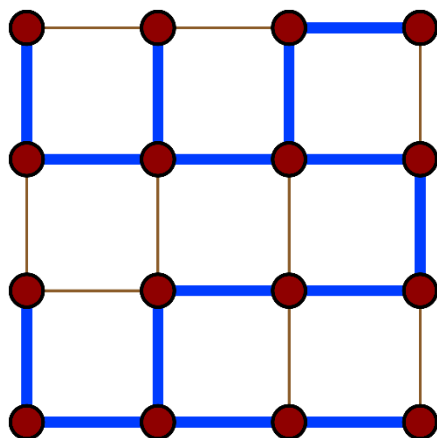
- 一般化された割り当て問題も最大流に帰着できる：
- 2部グラフの構成法：
 - v_s から U の各頂点に容量2の辺
 - U から v_t の各頂点に容量10の辺



全域木

全域木： グラフを木で近似

- グラフ G の全域木とは、 G の全頂点を含む部分グラフであって、かつ、木となっているもの
- 辺にコストがある場合は、コストの和が最小となるものを最小全域木という
 - グラフを木で近似したものとして利用できる



https://en.wikipedia.org/wiki/Spanning_tree

最小全域木の構成： 貪欲法

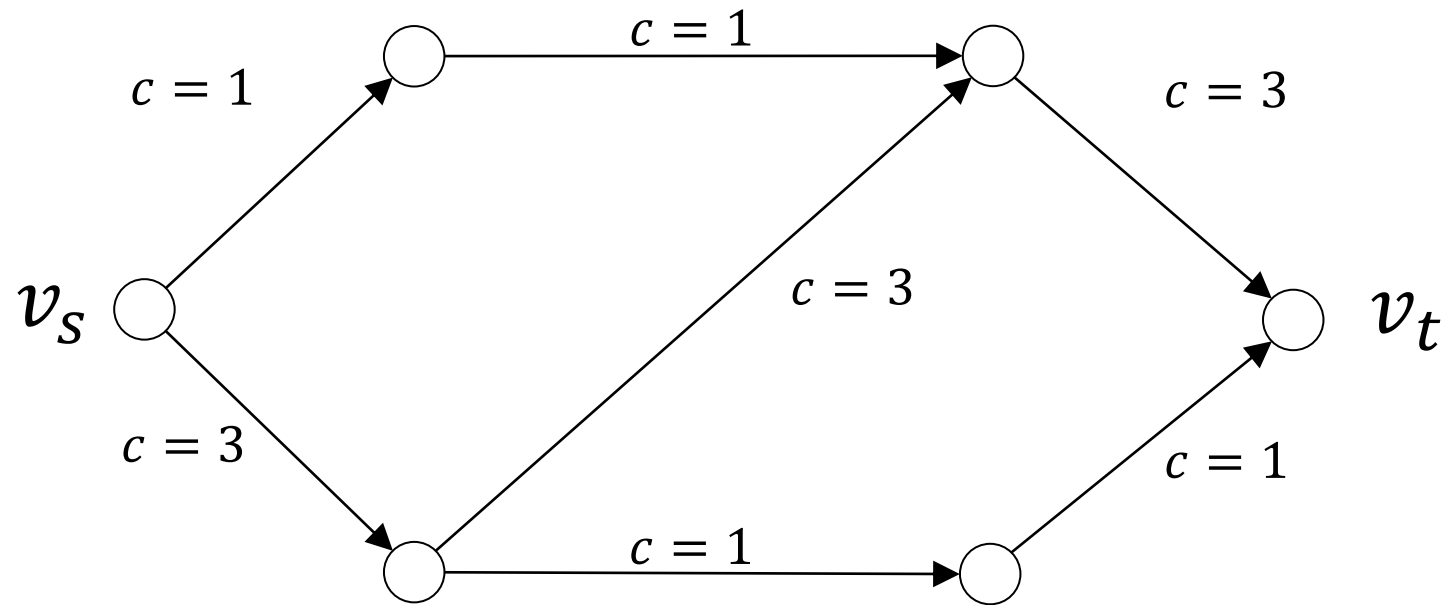
- 貪欲法：
 - 逐次的に解を構成する（辺をひとつずつ追加する）
 - 最も評価の高いものを選ぶ
（最もコストの小さい辺を選ぶ）
 - 一旦選んだものは変えない
- 最小全域木における貪欲法：
 - 最もコストの小さい辺を選んでいく
 - ただし、閉路ができないという条件（木でなくなるため）

クラスカル法： 貪欲法による最小全域木の構成

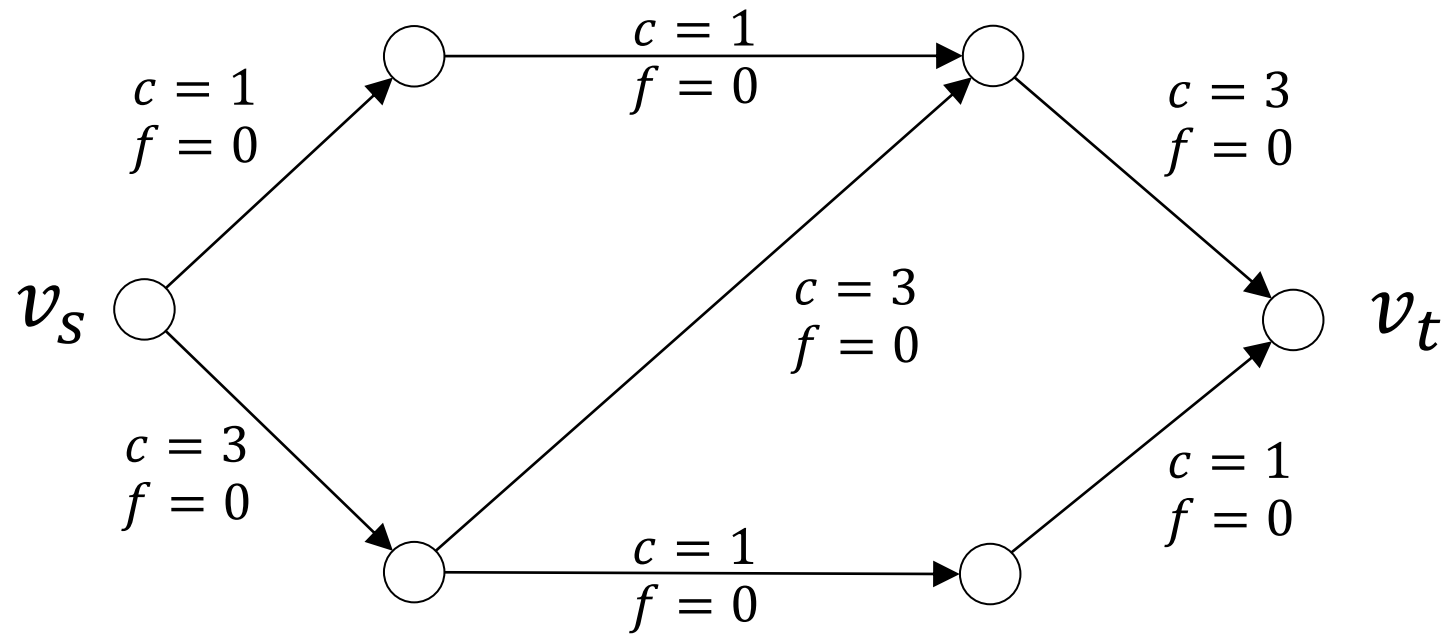
- 初期設定：全頂点を別々の（頂点1つの）木とする
- 各ステップで、2つの木を連結する（＝閉路をつくらない）
辺の中でコスト最小のものを選び、2つの木を統合する
 - － 辺が2つの木を連結するかどうかは、辺の両端の頂点がひとつの木に含まれるかどうかをチェックする
- 素集合データ構造：互いに素な部分集合を管理する
 - － Find操作：ある要素がどの部分集合に属するかを返す
 - － Union操作：2つの部分集合を1つにまとめる

付録：最大流問題の例

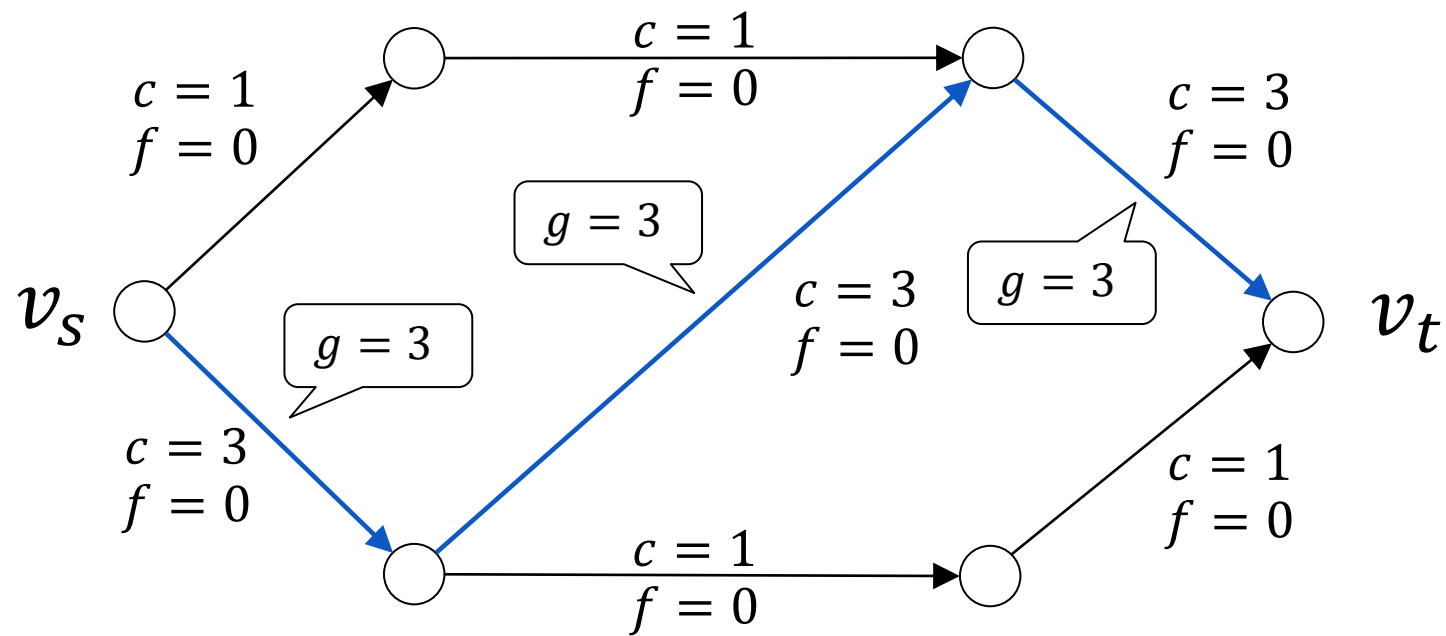
最大流問題： 例



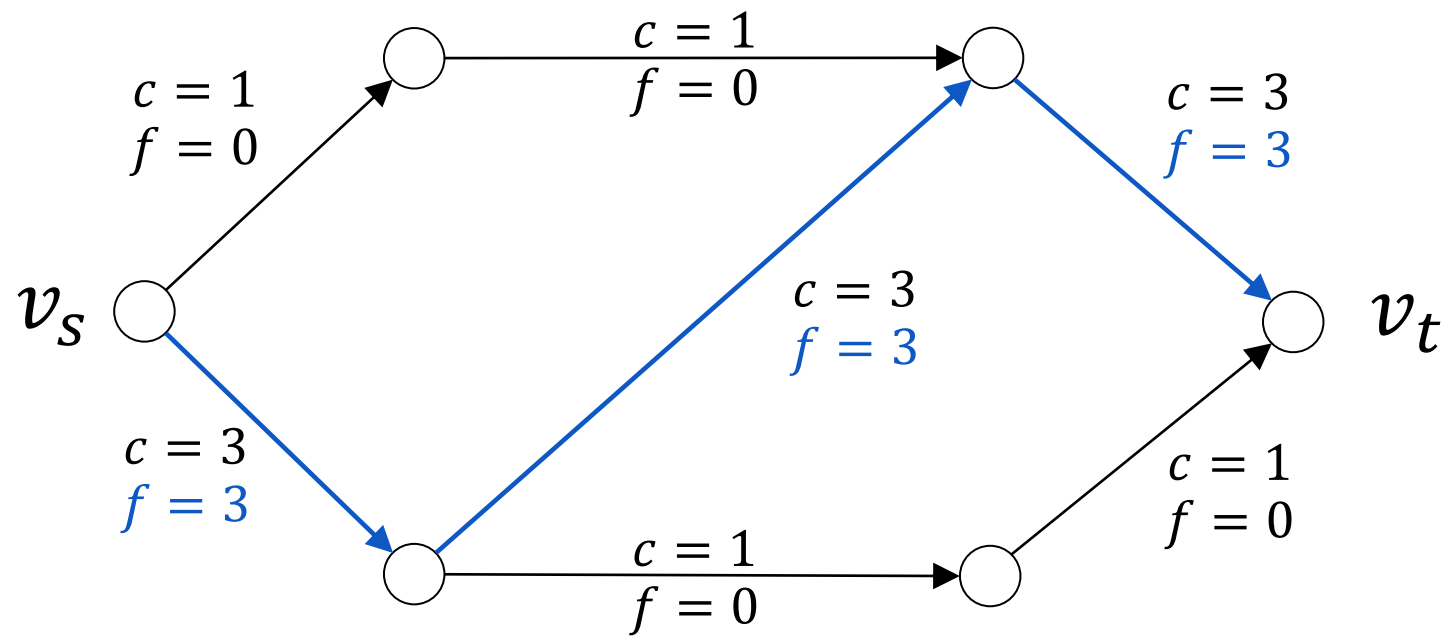
最大流問題： 例



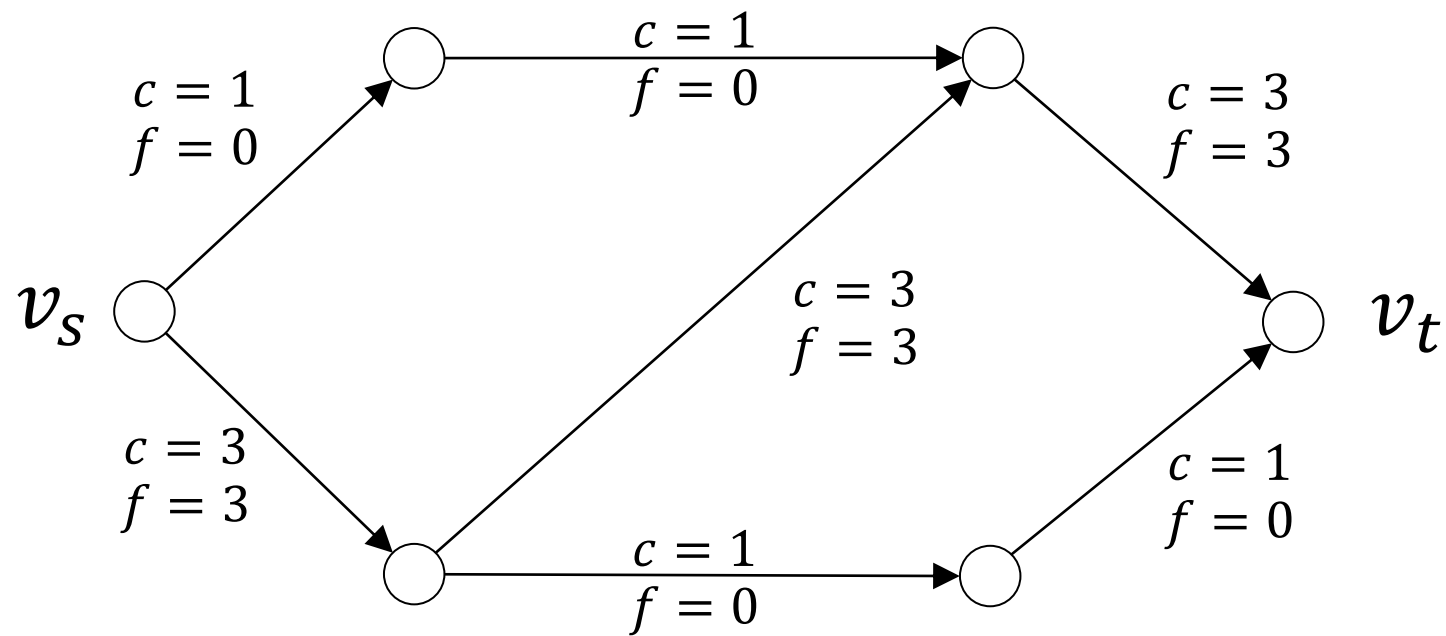
最大流問題： 例



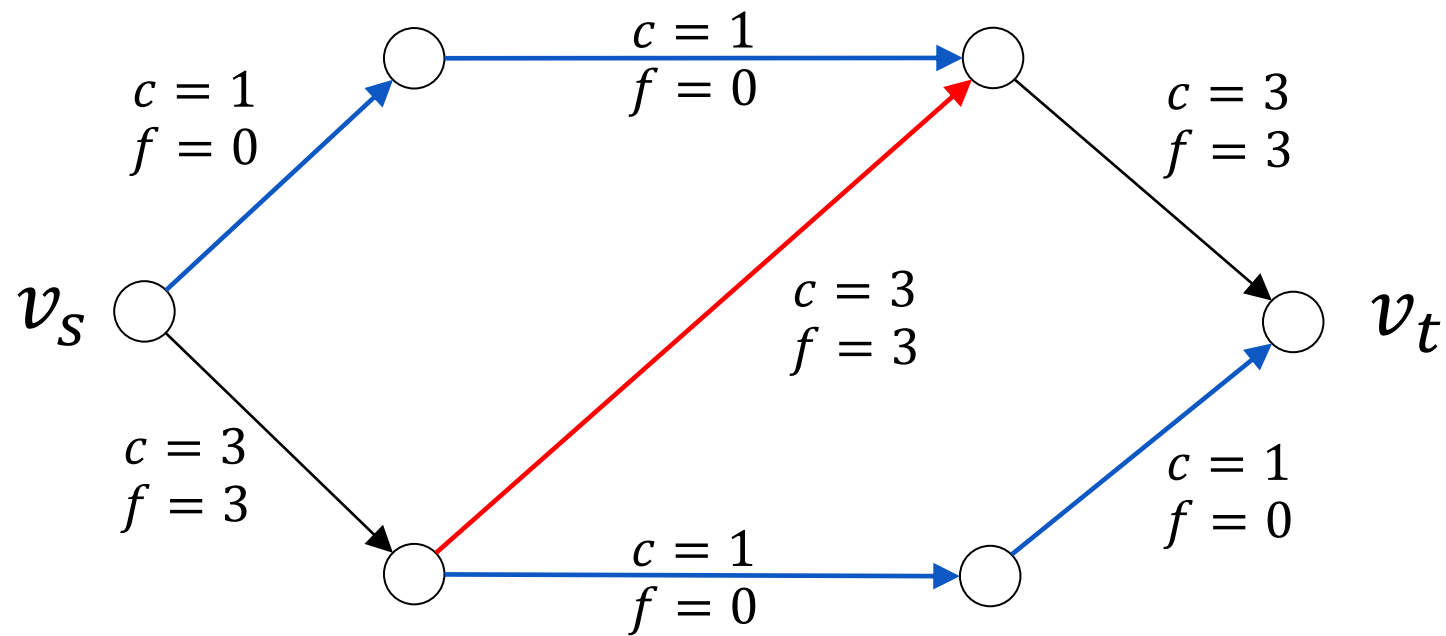
最大流問題： 例



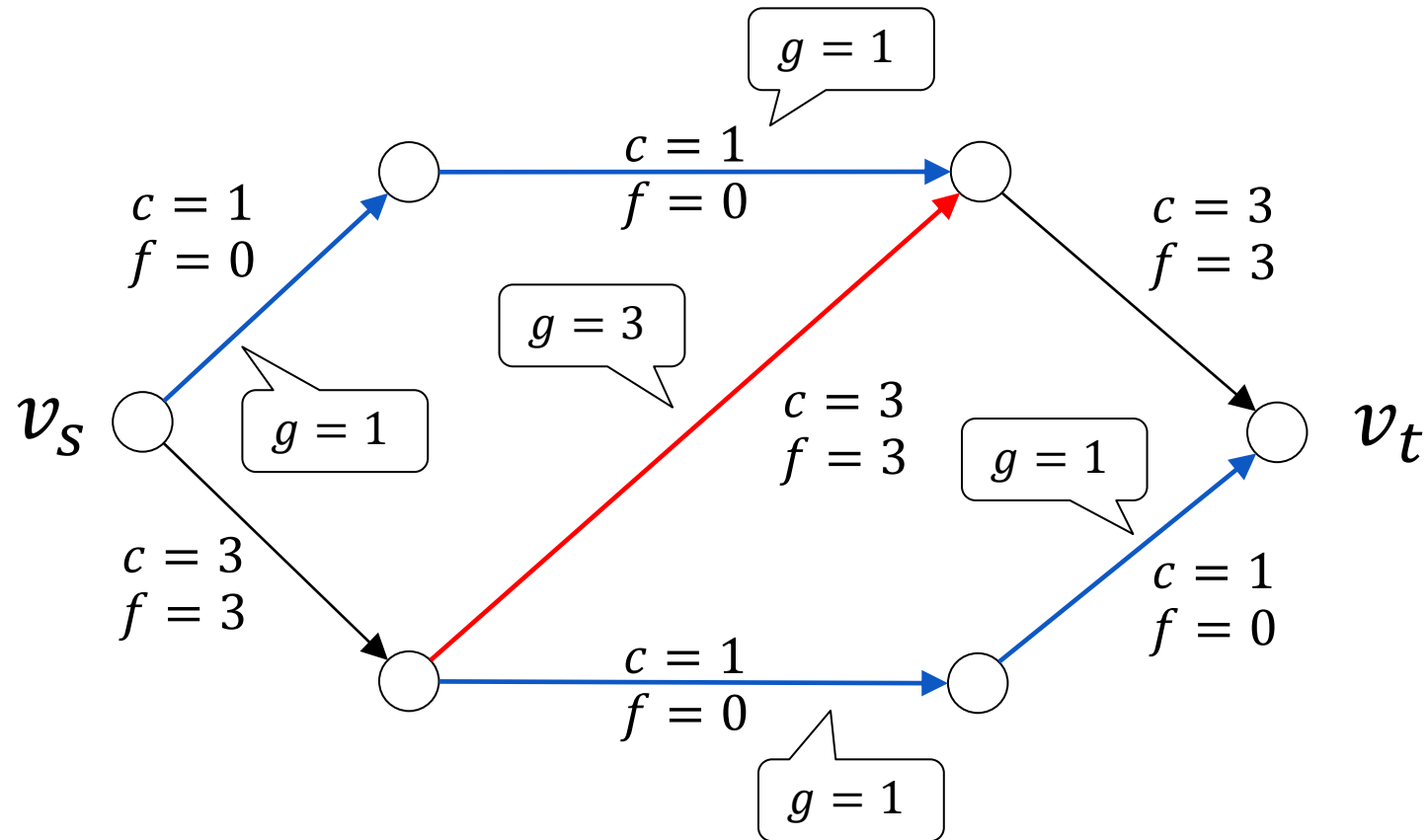
最大流問題： 例



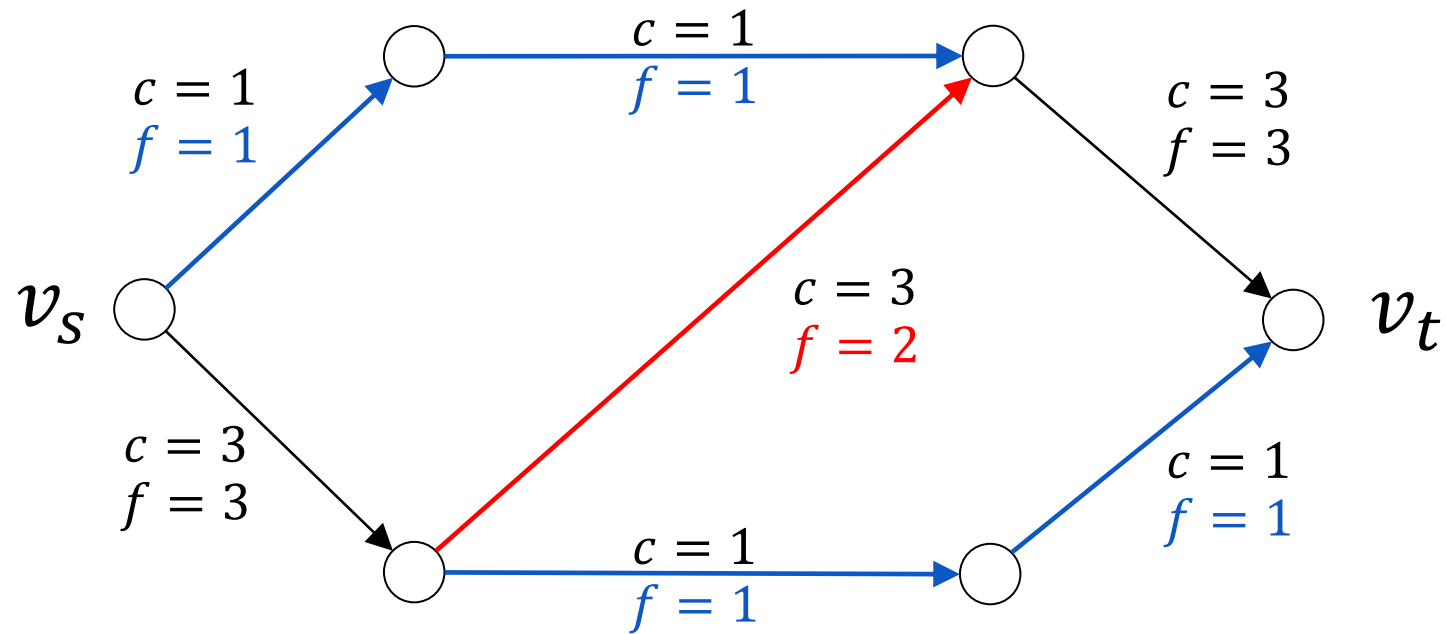
最大流問題： 例



最大流問題： 例



最大流問題： 例



最大流問題： 例

