機械学習ゼミ
劣モジュラ最適化と機械学習

Machine Learning
Machine Learning with Submodular Functions

2018/7/7 研究室ゼミ

福田研究室　修士2年
鈴木　新

目次

- 0章：記号の意味
- 1章：学習における劣モジュラ性
- 2章：劣モジュラ最適化の基礎
- 3章：劣モジュラ関数の最大化と貪欲法の適用
- 4章：最大流とグラフカット

※時間の都合上，内容めちゃ省略してます．
（気になる方は本を読んでください）

記号の意味

$S \cap T$ ・・・集合Sと集合Tの共通部分（黄）

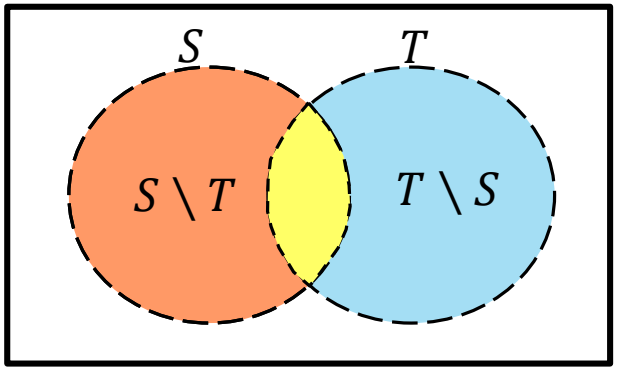
$S \cup T$ ・・・集合Sと集合Tの和集合（黄＋赤＋青）

$S \subseteq T$・・・集合Sは集合Tの部分集合

$i \in T$・・・ iは集合Tの要素

$\{i\}$・・・集合の要素i

$S \setminus T$・・・集合Sにあって集合Tにはない要素（赤）
ex) $S = \{1,2\}$, $\mathrm{T} = \{1,4,5\}$だったら$S \setminus T = 2$



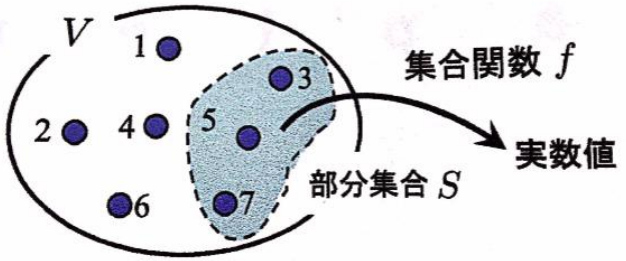
1.1はじめに

- 本書では、機械学習における組合せ的な側面を考える。
  - 組合せ・・・何らかの選択可能な集まりからその一部を選択すること
  - 機械学習でこの組合せが非常に重要となる
- 選択可能な対象・要素の集まりからその一部を選択する計算
  - ⇒集合関数（離散的な関数）の最適化につながる

集合関数とは

n個の選択可能な対象の数において
- 台集合：選択される対象の全体($V = \{1, \dots, n\}$)
- 集合関数：$f: 2^V \Rightarrow \mathbb{R}$

台集合7, 部分集合$S = \{3,5,7\}$



部分集合Sにおいて実数値を出力

1.1はじめに（集合関数の最適化）

アイスクリーム売上の需要予測を回帰分析で行いたい．しかし関係がない変数は選ばず有用なものを選びたい．

アイスクリーム売上の予測精度に関連する量がfで表されているとすれば，これを最大とするような項目の集合を求める問題になる．

もし，項目の台集合Vが「気温」と「料金」だけだったら組合せは

{気温}，{料金}，{気温，料金}，{なし}

の4通り，しかし現実はそんなことない（10個，100個，1000個あるかも）

10個・・・$2^{10} = 1024$通り

100個・・・$2^{100} = 1267650600228229401496703205376$通り

1000個・・・10715086071862673209484250490600018105614048117055336074437503883703510511249361224931983788156958581275946729175531468251871452856923140435984577574698574803934567774824230985421074605062371141877954182153046474983581941267398767559165543946077062914571196477686542167660429831652624386837205668069376通り

どれを採用するか，全パターンを列挙，判別することは不可能

1.1はじめに（集合関数の最適化）

このような現象のことを組み合わせ爆発という.

⇒計算機を用いて集合関数最適化などの組み合わせ計算を扱う難しさの根源になっている.

（フカシギお姉さんでYoutube調べるとわかりやすい動画出てきます．）
https://www.youtube.com/watch?v=Q4gTV4r0zRs

組み合わせ爆発を防ぐために様々なアルゴリズムが開発されている.

1.2劣モジュラ性の導入

劣モジュラ(関数)最適化

集合関数における凸性にあたる構造, かつ凹関数のような性質ももつ.
「集合関数最適化などの組み合わせ的な計算を扱う難しさ」に直面しないよう考える最適化.

集合Sと集合Tに関して

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T) \qquad (\forall S, T \subseteq V) \qquad (1.1)$$

この式↑を満たしていれば「劣モジュラ性を満たしている」といえる

組合せ最適化を考える際, 劣モジュラ性を満たすモデル(関数)を作成すれば, 組み合わせ爆発の問題を回避出来る可能性.

劣モジュラ構造を持つ離散最適化問題は，数理的枠組みの基盤が強固であり [室田 07]，潜在的に幅広い応用が展開可能なため，人工知能分野でも今後の重要な研究対象として注目できる．適用問題の例としては，多変量データから有用な変数集合を選択する問題，社会ネットワーク上の情報伝播で影響を最大にするノード集合を選択する問題などが知られている．また，ある種のコストや制約条件などを持つ複雑な問題に対しては，条件緩和や探索法の改良などにより，劣モジュラ最適化問題として定式化して解くための研究も最近活発に行われつつある．劣モジュラ最適化として定式化可能な新たな問題領域の探究も重要な研究課題と言える．

劣モジュラ関数最大化による大規模クラスタリング，2009，入月ら

1.2劣モジュラ性の導入

$$\boldsymbol{f(S)+f(T)\geq f(S\cup T)+f(S\cap T)} \quad (\forall \boldsymbol{S,T\subseteq V}) \quad (1.1)$$

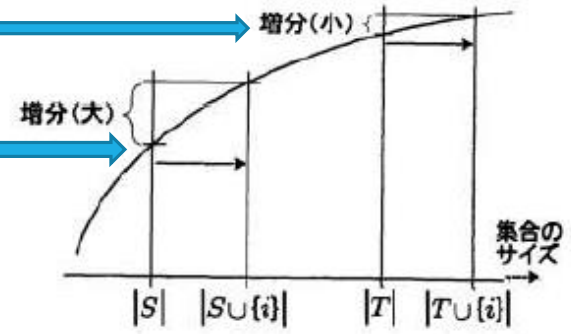
$S \subseteq T \subseteq V$かつ$i \subseteq V \setminus T$
$S' = S \cup \{i\}, \mathrm{T}' = \mathrm{T}$を代入すると

集合に要素{i}を加えた時の関数値の増分

$$f(S\cup\{i\})-f(S)\geq f(T\cup\{i\})-f(T) \quad (\forall S\subseteq T\subseteq V, \forall i\subseteq V\setminus T) \quad (1.2)$$

Tに包含される小さい集合Sに要素{i}を加えた時の関数値の増分

Sを包含する大きい集合Tに要素{i}を加えた時の関数値の増分



小さい集合に要素を加えた時の方が，大きい集合に要素を加えた時よりも影響が大きい

（詳細は3章で）

1.2劣モジュラ性の導入

ざっくりいうと

（単調）劣モジュラ性とは…
　…1個ずつ要素を追加した際に徐々に影響力が小さくなること！

ゲーマー（ゲームたくさん持ってる人）より初心者（ゲーム持ってない人）にゲームソフト一個あげた方が喜びは大きい

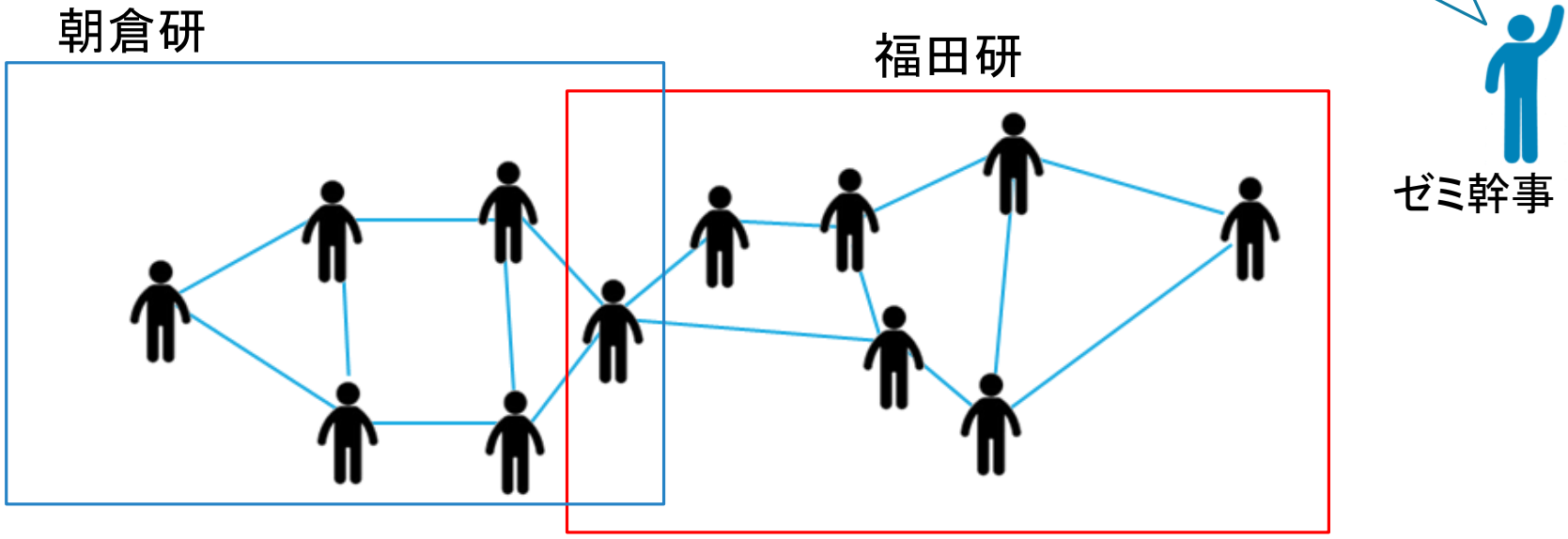
チケット何枚も持ってる人より，一枚も持ってない人にチケットあげた方が喜ぶ

https://www.youtube.com/watch?v=Z7eMzSHGGAE&t=339s

1.2劣モジュラ性の導入

劣モジュラ性とは1個要素を追加した際に徐々に影響力が小さくなる
・口コミ効果は同じグループ内で口コミを発信した人が少ないほど大きい

●来週の合同ゼミがなくなりました，誰に話すのが効果的？
（コミュ障なので二人までにしか流せません）

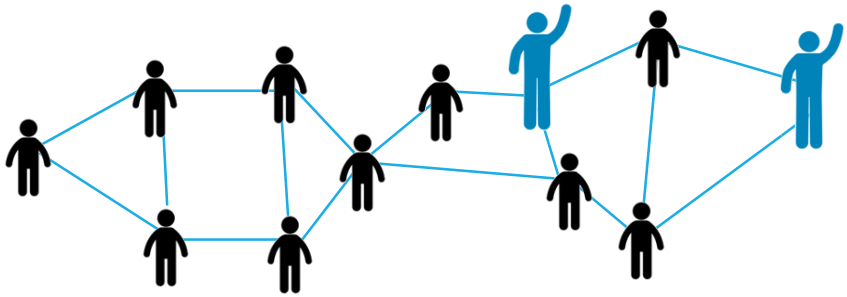


1日経つとその人の隣の人に伝わる場合，誰に話すのが効果的？

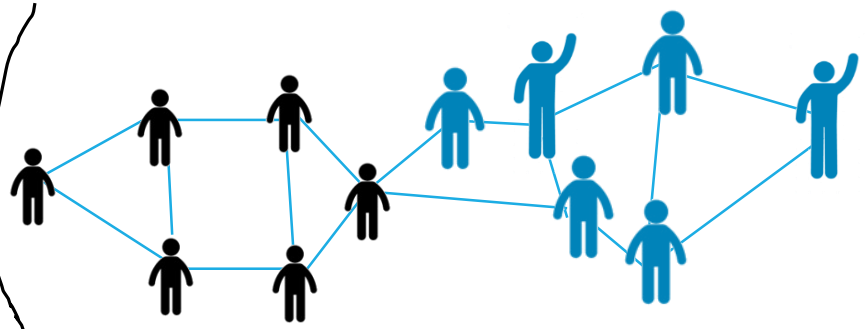
1.2劣モジュラ性の導入

劣モジュラ性とは1個要素を追加した際に徐々に影響力が小さくなる
・口コミ効果は同じグループ内で口コミを発信した人が少ないほど大きい

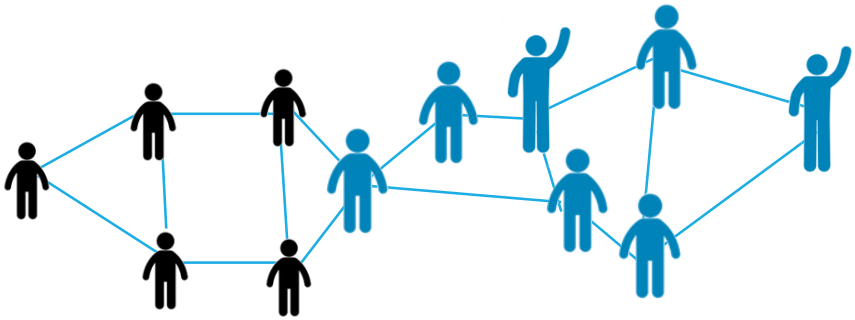
二人とも福田研に話した場合



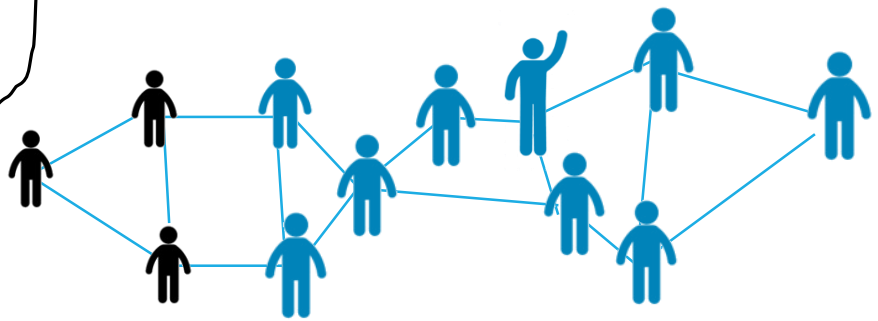
1日目



2日目



3日目

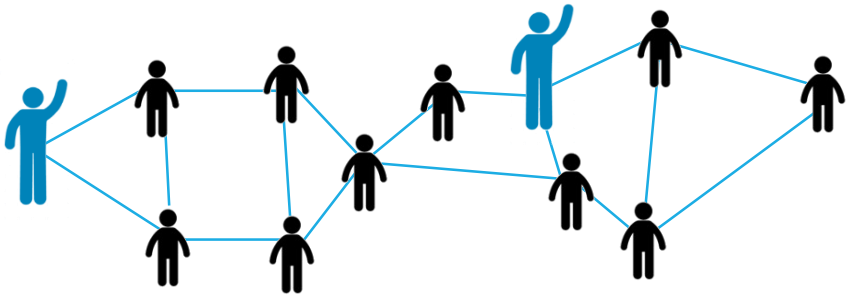


4日目（朝倉研で伝わってない人がいる…）

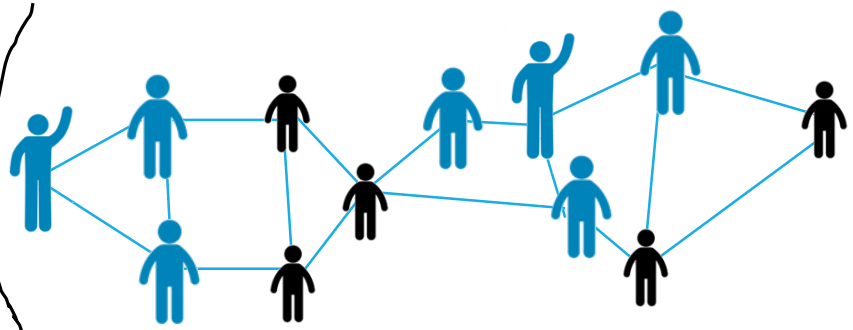
1.2劣モジュラ性の導入

劣モジュラ性とは1個要素を追加した際に徐々に影響力が小さくなる
・口コミ効果は同じグループ内で口コミを発信した人が少ないほど大きい

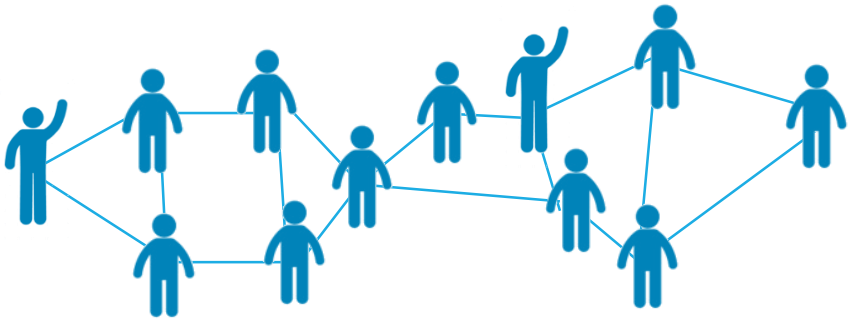
福田研，朝倉研それぞれ一人の場合



1日目



2日目



3日目

3日目でコンプリート！
この差が劣モジュラ性！

同じ集合(研究室)内で要素増やすよりも他集合で増やした方が効果的（同集合で増やしても効果薄）

1.3機械学習における劣モジュラ性

- 劣モジュラ関数の例
  - カット関数
  - カバー関数

- 劣モジュラ関数最大化を応用した研究も多い
  - 能動学習
  - ノンパラメトリック・ベイズ推定
  - インフルエンサーへのオンライン広告問題

- 離散的なデータ構造を表すのにも適している
  - コンピュータービジョン分野

2. 劣モジュラ最適化の基礎

凸関数最小化・凹関数最大化は最適化問題として扱いやすい

劣モジュラ関数最適化において

- 劣モジュラ関数の背後には凸性があるため，最小化問題は多項式時間で解ける
- 劣モジュラ関数の最大化問題はほとんどの問題設定においてNP困難であるが，劣モジュラ関数の背後には凹性のような性質があるため，かなり良い近似解を多項式時間で計算できる

多項式時間＝ n^c
⇔指数時間＝c^n
(cは定数)

現実的な時間で解を求めることができない

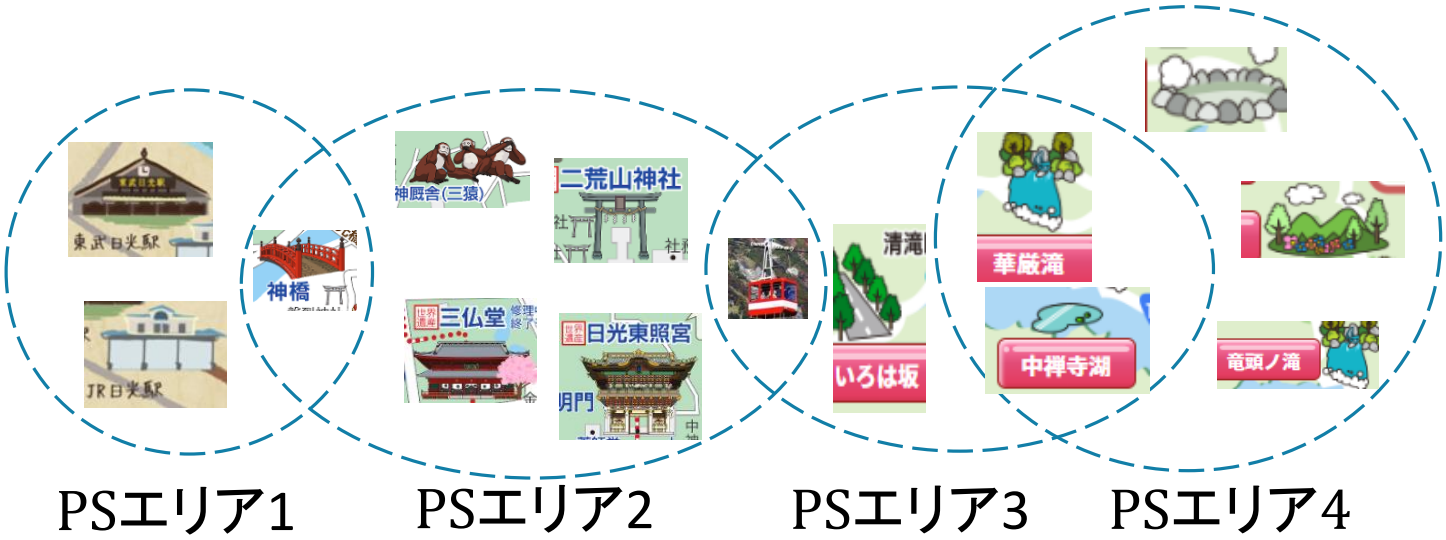
この章では劣モジュラ関数の紹介と，軽く証明を行います
（カバー関数とカット関数，実際に最適化するのは次章以降）

http://www.nikkou-sanraku.jp/introduction/index01.html

http://www.homemate-research.com/tourism_map/09/nikkoshi.shtml
https://mapandnews-japan.com/09nikkotoshogu/demomobile.html?csv=poi.csv

2.1.1劣モジュラ関数の具体例(カバー関数)

ある観光地に複数の観光スポットがある.
Wi-Fiパケットセンサー(PS)を4台(有限集合$V = \{1,2,3,4\}$)おける時



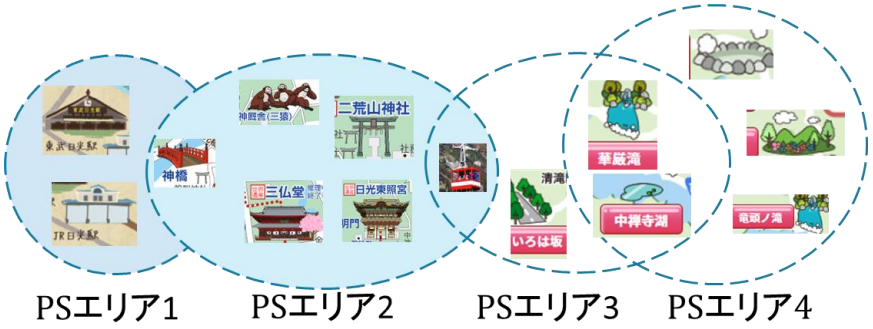
パケットセンサーを置くと点線円内の観光スポットをカバーできる.
この時, パケットセンサーを置く場所の集合を$S \subseteq V$として, 関数は

$$f_{cov}(S) = \begin{pmatrix} S\text{に対応するパケットセンサーが} \\ \text{カバーする観光スポットの数} \end{pmatrix}$$

2.1.1劣モジュラ関数の具体例(カバー関数)

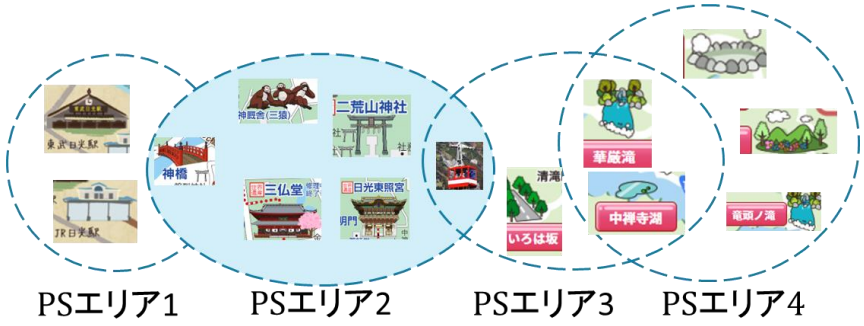
例えば，$S = \{1\},\{2\},\{1,2\}$の時，
$f_{cov}(S)$はそれぞれ，

- エリア1に置いたとき
$$f_{cov}(\{1\}) = 3$$
- エリア2に置いたとき
$$f_{cov}(\{2\}) = 6$$
- エリア1,2に置いたとき
$$f_{cov}(\{1,2\}) = 8$$



PSエリア1,2にWi-FiPSおいたとき

PSエリア1にWi-FiPSおいたとき



PSエリア2にWi-FiPSおいたとき

2.1.1劣モジュラ関数の具体例(カバー関数)

- 範囲が重なっているエリアがあるため，線形性は成り立たない

$$f_{cov}(\{1\}) + f_{cov}(\{2\}) \neq f_{cov}(\{1,2\})$$

- 今回，Wi-FiPSを置ける場所は4か所なので，Wi-FiPS設置個所の部分集合の組み合わせは

$$2^4 = 16\text{通り}$$

- この関数(カバー関数)の劣モジュラ性を証明(のイメージ)．
  重複部分がある
  ⇒新しいWi-FiPSを置けば置くほど，設置する効果は減少してく．

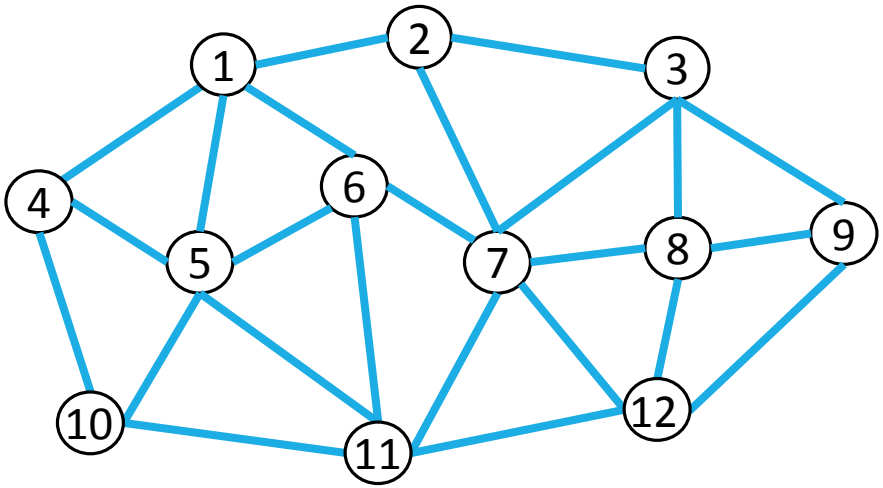
$$\underbrace{f(\{1\} \cup \{2\})}_{8} - \underbrace{f(\{1\})}_{3} \geq \underbrace{f(\{1,2\} \cup \{3\})}_{11} - \underbrace{f(\{1,2\})}_{8}$$

$$f(S \cup \{i\}) - f(S) \geq f(T \cup \{i\}) - f(T)$$
$$(\forall S \subseteq T \subseteq V, \forall i \subseteq V \setminus T) \quad (1.2)$$

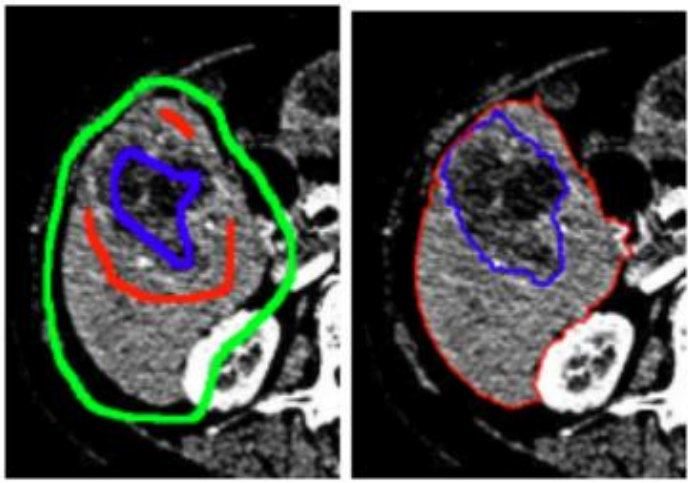
2.1.2劣モジュラ関数の具体例(グラフカット関数)

グラフカット関数

ネットワークやグラフに関する最適化において重要な役割を果たす劣モジュラ関数．無向グラフと有向グラフの2種類のパターンがある．
⇒情報処理の画像処理などに使われる．



無向グラフのネットワーク
矢印あれば有向グラフ



CT画像から肝臓の部分だけ取り出したい

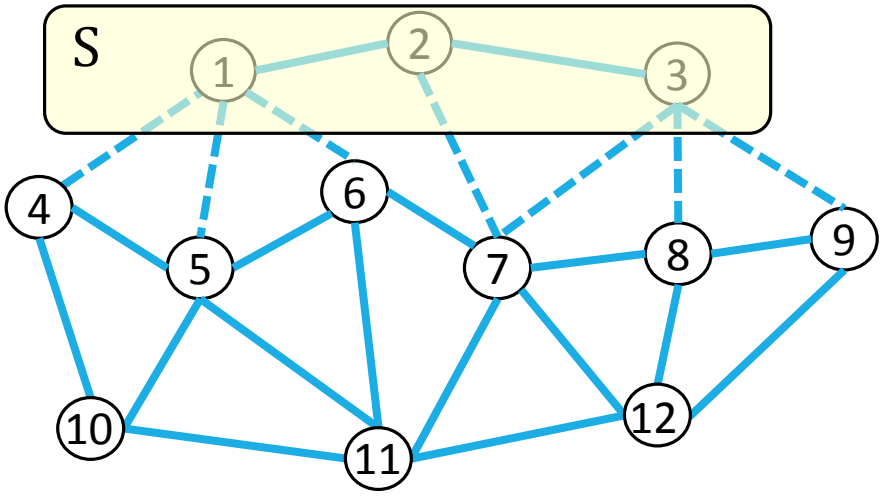
http://zellij.hatenablog.com/entry/20131004/p1

2.1.2劣モジュラ関数の具体例(グラフカット関数)

無向グラフの場合

12個の頂点からなる頂点集合$\nu = \{1, \dots, 12\}$と(向きのない)枝集合εが与えられている.

頂点の部分集合$S \subseteq \nu$について, 枝の一端がS, もう一端が$\nu \setminus S$に含まれるような枝集合εの部分集合を$\delta(S)$. $|\delta(S)|$を枝の本数とする.



頂点の部分集合

$$S = \{1,2,3\}$$

の時, 枝の部分集合は左図の点線の部分, つまり

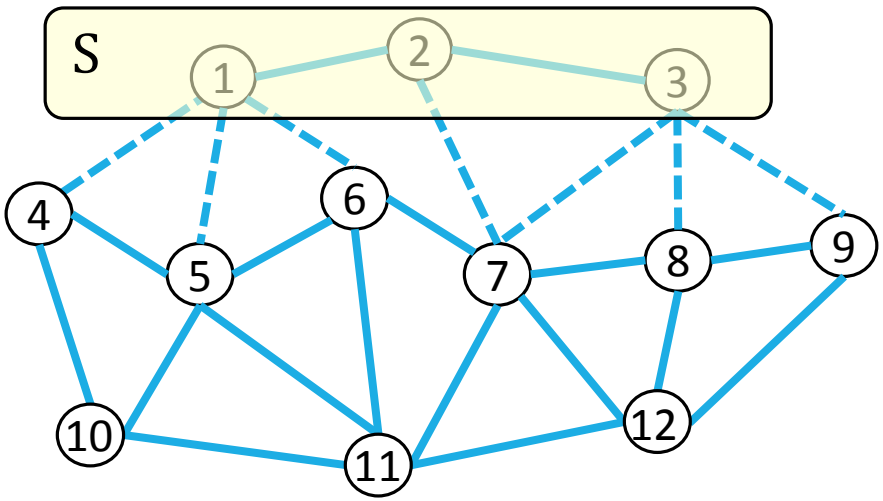
$$|\delta(\{1,2,3\})| = 7$$

となる.

$$|\delta(\{1\})| = 4$$
$$|\delta(\{1,2\})| = 5$$
$$|\delta(\{1,3\})| = 8$$

2.1.2劣モジュラ関数の具体例(グラフカット関数)

この関数(カバー関数)の劣モジュラ性のチェック.



頂点の部分集合

$$S = \{1,2,3\}$$

の時, 枝の部分集合は左図の点線の部分, つまり

$$|\delta(\{1,2,3\})| = 7$$

となる.

$$|\delta(\{1\})| = 4$$
$$|\delta(\{1,2\})| = 5$$
$$|\delta(\{1,3\})| = 8$$

$$\underbrace{|\delta(\{1,2\})|}_{5} + \underbrace{|\delta(\{1,3\})|}_{8} \geq \underbrace{|\delta(\{1,2,3\})|}_{7} + \underbrace{|\delta(\{1\})|}_{4}$$

劣モジュラ性を満たす.(他のパターンでやってみても満たすはず…)

2.1.2劣モジュラ関数の具体例(グラフカット関数)

有向グラフの時はどうなるか

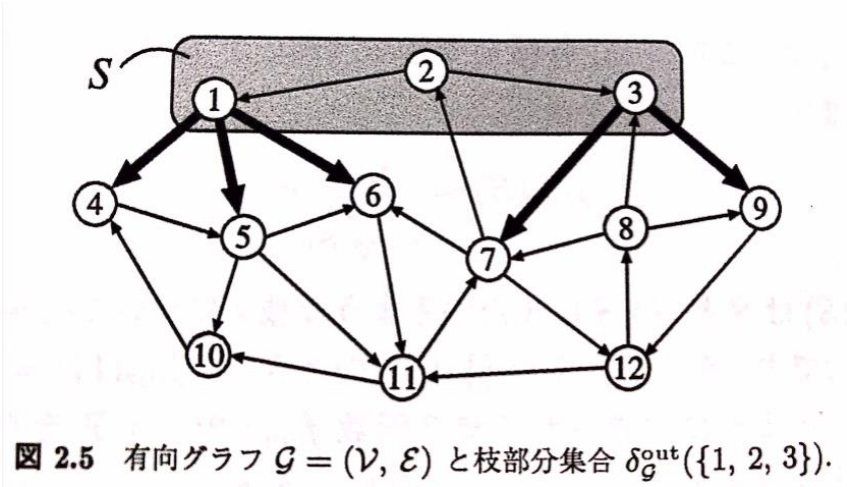


図 2.5　有向グラフ $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ と枝部分集合 $\delta_{\mathcal{G}}^{\mathrm{out}}(\{1, 2, 3\})$.

頂点の部分集合

$$S = \{1,2,3\}$$

の時，枝の部分集合は浸り図の点線の部分，つまり

$$|\delta_d(\{1,2,3\})| = 5$$

となる.

$$|\delta_d(\{1\})| = 3$$
$$|\delta_d(\{1,2\})| = 4$$
$$|\delta_d(\{1,3\})| = 5$$

$$\underbrace{|\delta_d(\{1,2\})|}_{4} + \underbrace{|\delta_d(\{1,3\})|}_{5} \geq \underbrace{|\delta_d(\{1,2,3\})|}_{5} + \underbrace{|\delta_d(\{1\})|}_{3}$$

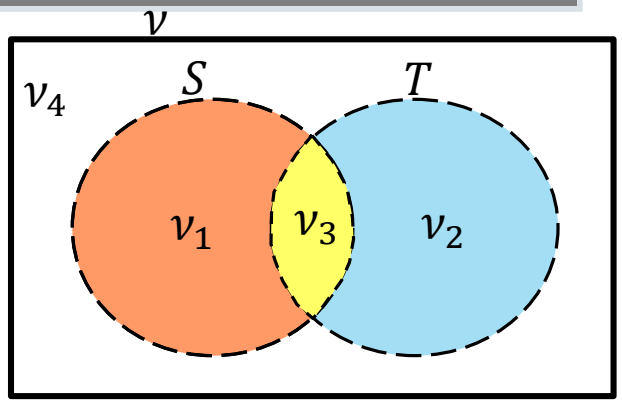
劣モジュラ性を満たす.

⇒一般式に直してざっくり証明してみよう

2.1.2劣モジュラ関数の具体例（グラフカット関数）

任意に与えられた頂点部分集合を$S, T \subset \nu$は4つに分けられる.

$$\nu_1 = S \setminus T \qquad \nu_2 = T \setminus S$$
$$\nu_3 = S \cap T \qquad \nu_4 = \nu \setminus (S \cup T)$$
$$\nu = \nu_1 \cup \nu_2 \cup \nu_3 \cup \nu_4$$



これより,

$$S = \nu_1 \cup \nu_3, T = \nu_2 \cup \nu_3, S \cup T = \nu_1 \cup \nu_2 \cup \nu_3$$

ν_aから出てν_bに入るような全ての枝の枝容量の総和を$C_{a,b}$と置くと.

$$|\delta(S)| = C_{1,2} + C_{1,4} + C_{3,2} + C_{3,4}$$
$$|\delta(T)| = C_{2,1} + C_{2,4} + C_{3,1} + C_{3,4}$$
$$|\delta(S \cup T)| = C_{1,2} + C_{1,4} + C_{3,2} + C_{3,4}$$
$$|\delta(S \cap T)| = C_{1,2} + C_{1,4} + C_{3,2} + C_{3,4}$$

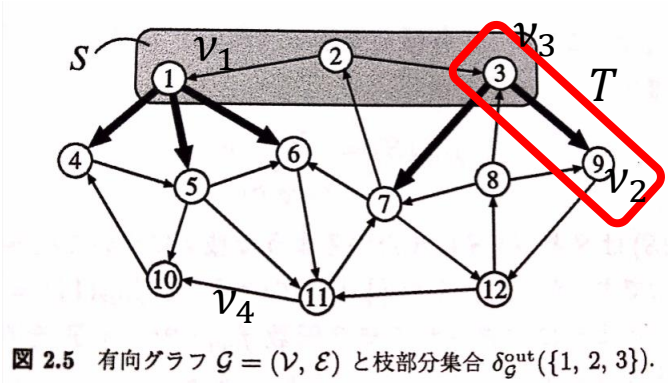


図 2.5 有向グラフ $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ と枝部分集合 $\delta_{\mathcal{G}}^{\text{out}}(\{1, 2, 3\})$.

$\nu_1 = \{1,2\}, \nu_2 = \{9\},$
$\nu_3 = \{3\}, \nu_4 = \{1,2,3,9\text{以外}\},$

$$|\delta(S)| + |\delta(T)| - |\delta(S \cup T)| - |\delta(S \cap T)| = C_{1,2} + C_{2,1} \geq 0$$

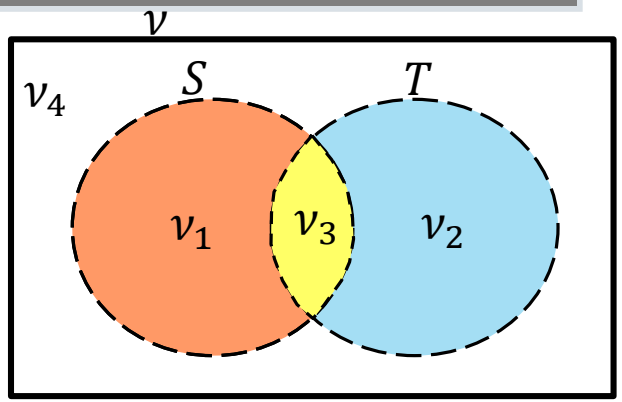
2.1.2劣モジュラ関数の具体例（グラフカット関数）

ν_aから出てν_bに入るような全ての枝の枝容量の総和を$C_{a,b}$と置くと.

$$|\delta(S)| = C_{1,2} + C_{1,4} + C_{3,2} + C_{3,4}$$
$$|\delta(T)| = C_{2,1} + C_{2,4} + C_{3,1} + C_{3,4}$$
$$|\delta(S \cup T)| = C_{1,2} + C_{1,4} + C_{3,2} + C_{3,4}$$
$$|\delta(S \cap T)| = C_{1,2} + C_{1,4} + C_{3,2} + C_{3,4}$$

$$|\delta(S)| + |\delta(T)| - |\delta(S \cup T)| - |\delta(S \cap T)| = C_{1,2} + C_{2,1} \geq 0$$

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T)$$



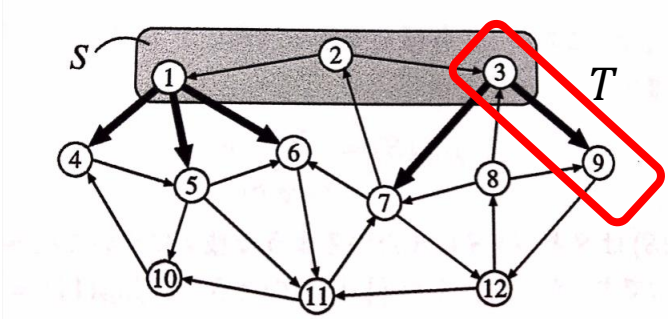


図 2.5　有向グラフ $G = (V, E)$ と枝部分集合 $\delta_G^{out}(\{1, 2, 3\})$.

$\nu_1 = \{1,2\}, \nu_2 = \{9\},$
$\nu_3 = \{3\}, \nu_4 = \{1,2,3,9以外\},$

3. 劣モジュラ最大化と貪欲法

3章：劣モジュラ最大化と貪欲法

劣モジュラ関数最大化において

- 劣モジュラ関数の最大化問題はほとんどの問題設定においてNP困難であるが，劣モジュラ関数の背後には凹性のような性質があるため，かなり良い近似解を多項式時間で計算できる

主に単調劣モジュラ関数最大化について考えます

単調劣モジュラ関数最大化

目的：$f(S) \Rightarrow$ 最大

制約：$S \subseteq V, |S| \leq k$

NP困難な問題⇒厳密な最適解を多項式時間で求めることは困難

3.1 カバー関数最大化問題

ある観光地に複数の観光スポットがある.
Wi-Fiパケットセンサー(PS)を2台（有限集合V = {1,2,3,4}）おける時
どのエリアに置けば多くの観光地のデータ取れる？

PSエリア1　PSエリア2　PSエリア3　PSエリア4

⇒まずは全組み合わせを考えてみる

$$f_{cov}(\{1\}) = 3, f_{cov}(\{2\}) = 6, f_{cov}(\{3\}) = 4, f_{cov}(\{4\}) = 5,$$
$$f_{cov}(\{1,2\}) = 8, f_{cov}(\{1,3\}) = 7, f_{cov}(\{1,4\}) = 8, f_{cov}(\{2,3\}) = 9,$$
$$f_{cov}(\{2,4\}) = 11, f_{cov}(\{3,4\}) = 8,$$

3.1.2カバー関数最大化問題 近似アルゴリズム（貪欲法の導入）

全組み合わせを考えると，$f_{cov}(\{2,4\}) = 11$が最大となる
⇒組み合わせ数が多くなると，列挙しきれなくなる

近似アルゴリズムを用いる
⇒アルゴリズムによって近似解が多項式時間に得られる.
⇒最適値と近似解の目的関数の比がどんな場合でも一定値より良くなることが証明されている.

※近似率α
α ∗ (問題の最適値) ≤ (近似アルゴリズムが出力する解)

カバー関数の場合，貪欲法を用いる（近似率は0.63)
0.63 ∗ (問題の最適値) ≤ (貪欲法が出力する解)

3.1カバー関数最大化問題 近似アルゴリズム(貪欲法の導入)

Step0. Sを空集合{}とする.

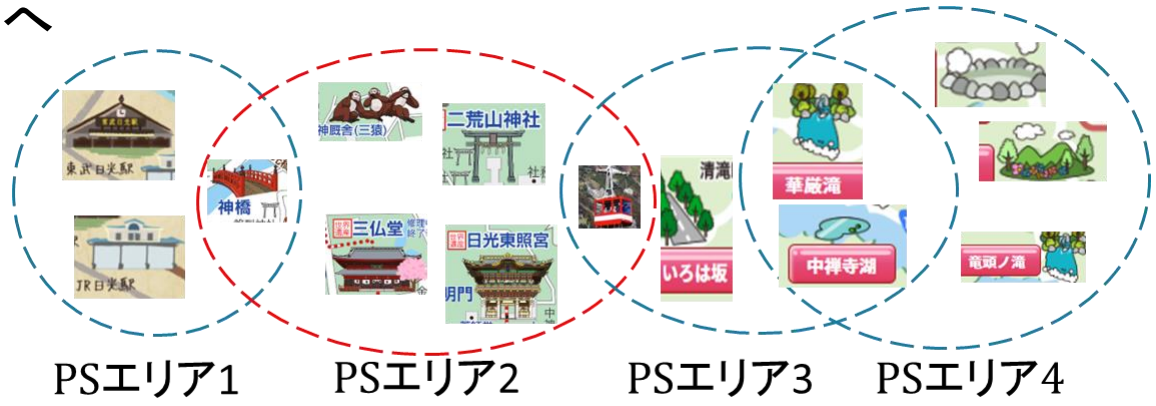
Step1. $|S| = k$ならば$S_{GA} := S$を出力し終了. $|S| \neq k$ならばstep2へ

Step2. $f(S \cup \{i\})$を最大化する$i \in V \setminus S$を1つ選び,
$S \cup \{i\} = S$と置きなおす.

今回の場合集合S はWi − FiPSの設置場所, kはWi − FiPSの数,

Step1.
$|\{\}| = 0 < 2$より, step2.へ

Step2.
$i = 3$がfを最大化.
$S = \{2\}$となる.



3.1カバー関数最大化問題 近似アルゴリズム(貪欲法の導入)

Step0. Sを空集合{}とする.

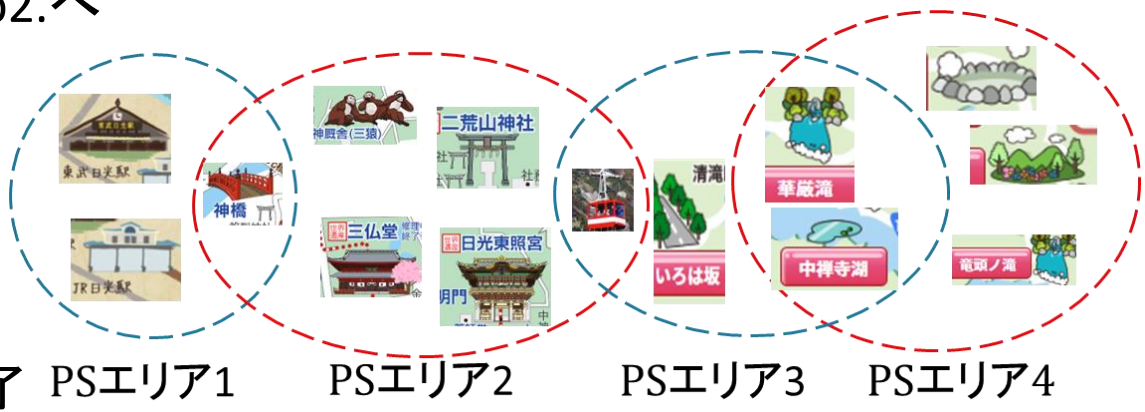
Step1. $|S| = k$ならば$S_{GA} := S$を出力し終了. $|S| \neq k$ならばstep2へ

Step2. $f(S \cup \{i\})$を最大化する$i \in V \setminus S$を1つ選び,
$S \cup \{i\} = S$と置きなおす.

今回の場合集合S はWi − FiPSの設置場所, kはWi- FiPSの数,

Step1.
$|\{2\}| = 1 < 2$より, step2.へ

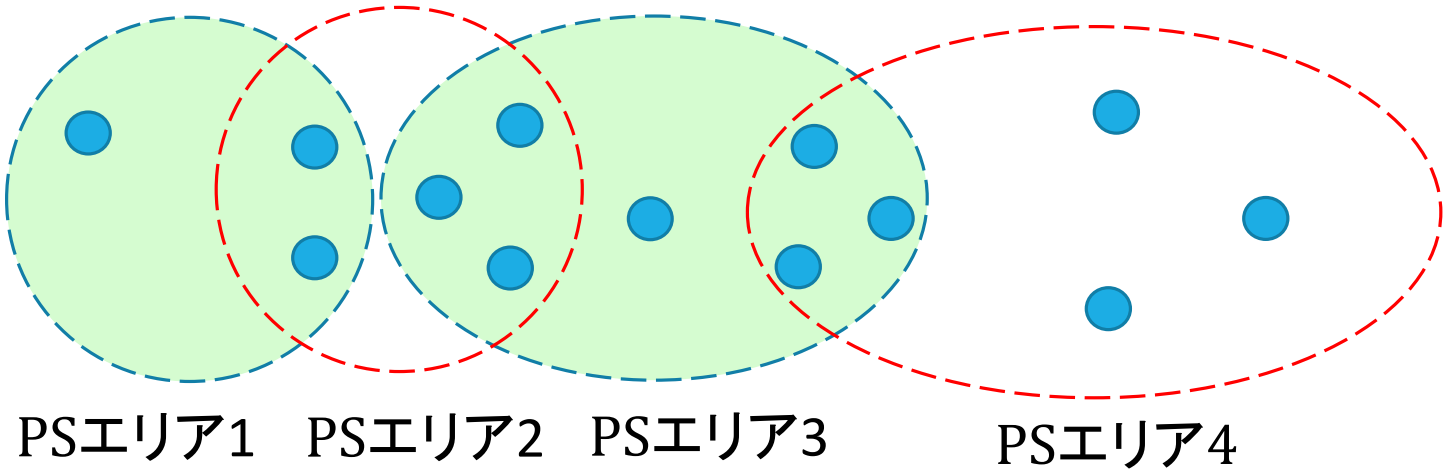
Step2.
$i = 4$がfを最大化.
$S = \{2,4\}$となる.



Step1.
$|\{3,4\}| = 2 = k$より終了
出力解は11となる.

3.1カバー関数最大化問題
近似アルゴリズム（貪欲法の導入）

ある観光地に複数の観光スポット● がある.
Wi-Fiパケットセンサー(PS)を2台（有限集合V = {1,2,3,4} ）おける時
どのエリアに置けば多くの観光地のデータ取れる？



$f(\{P_2, P_4\})$=11が最適解，貪欲法で解くと $f(\{P_3, P_1\})$=10に
⇒0.63近似は満たしている.

3.3カバー関数（センサ配置問題の活用）

・Wi-FiPSは無限には置けない

・実際は計測の精度なども絡んでくる.

⇒近くにWi-FiPS複数ある場合は観測誤差が小さい

観測誤差を小さくなるようWi-FiPSを配置する問題になったり

4.1.1.最大流とグラフカット

4章：グラフカットと最大流

多項式時間＝ n^c
⇔指数時間＝c^n
(cは定数)

劣モジュラ関数最小化において

- 劣モジュラ関数の背後には凸性があるため，最小化問題は多項式時間で解ける

一般の劣モジュラ関数・・・$\mathcal{O}(n^5EO + n^6)$←最悪計算量

⇒要素数多い計算，反復計算にはまだ不十分

個々の扱う問題で十分な表現力の範囲で高速に最小化可能な劣モジュラ関数の特殊クラスを考える

4.1.1.最大流とグラフカット

- グラフカット
  - コンピュータービジョンで頻繁に扱われる
  - 数万～数百万オーダーの問題に対しても実用的時間で適用可能なアルゴリズム

- 劣モジュラの視点から見ると，カット関数（劣モジュラ性あり）の最小化を行っている.

カット関数の最小化

最大流アルゴリズム

（理論的・実用的に高速なアルゴリズムの適用が可能）

4.1.1.カットとフロー

有向グラフ$\mathcal{G}(\mathcal{V},\mathcal{E})$上で,

<u>頂点集合$\mathcal{V}=\{s\}\cup\{t\}\cup V$</u>

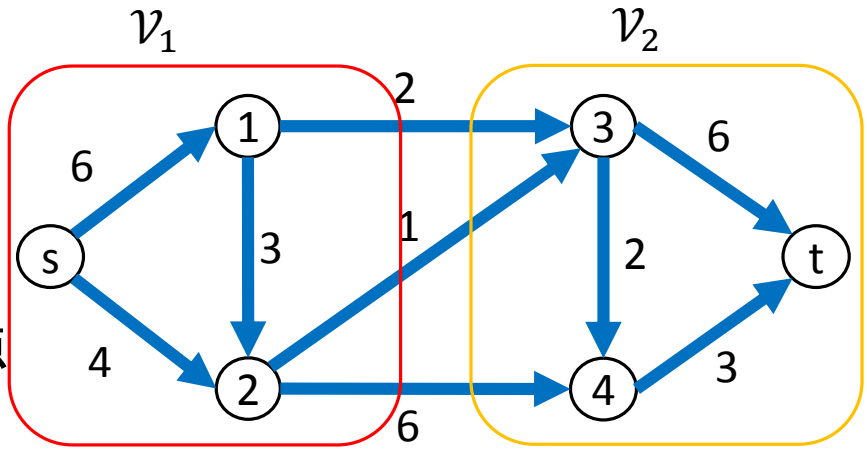
$\{s\}$:ソース(特別な頂点)

$\{t\}$:シンク(sとは異なる特別な頂点)

$V=\{1,..,n\}$:それ以外のn個の頂点

<u>枝集合$\mathcal{E}$</u>

各枝$e\in\mathcal{E}$には正の枝容量c_eが含まれている.



$V=\{1,2,3,4\}$,頂点数6, 枝数9の場合

$s-t$カット

有向グラフ$\mathcal{G}(\mathcal{V},\mathcal{E})$を順序付きの分割$(\mathcal{V}_1,\mathcal{V}_2)$のうち, $s\in\mathcal{V}_1, t\in\mathcal{V}_2$を満たすもの. Vの任意の部分集合Sを用いて,

$$(\{s\}\cup S,\{t\}\cup(V\setminus S))$$

4.1.1.カットとフロー

$\boldsymbol{s-t}$カットについて，その容量は

$$\kappa_{s-t}(S) = \sum_{e \in \delta_G^{out}(\{s\} \cup S)} c_e$$

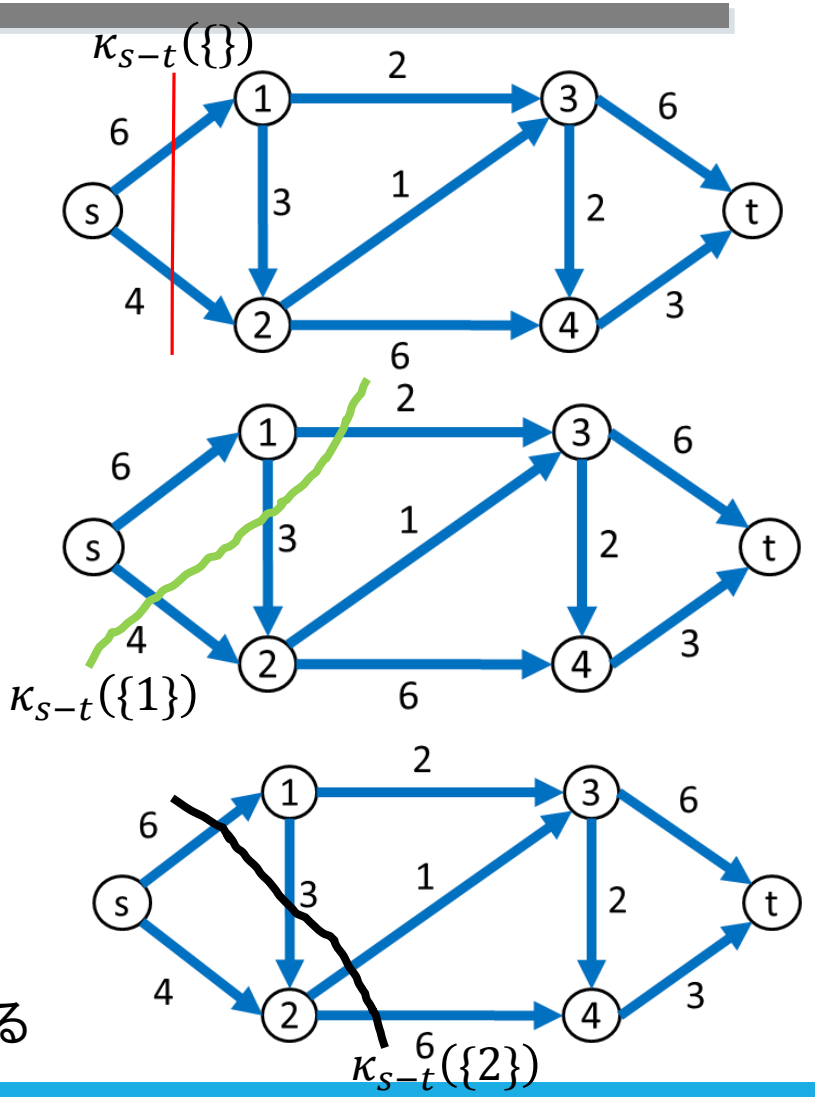
と定義
この関数をs-tカット関数と呼ぶ

右図でいうと

$$\kappa_{s-t}(\{\}) = c_{s,1} + c_{s,2} = 10$$
$$\kappa_{s-t}(\{1\}) = c_{1,2} + c_{1,3} + c_{s,2} = 9$$
$$\kappa_{s-t}(\{2\}) = c_{s,1} + c_{2,3} + c_{2,4} = 13$$

$\kappa_{s-t}(S)$を最小にするような
$\boldsymbol{s-t}$カットを求める.

⇒最大流アルゴリズムを用いると高速に解ける



4.1.1.フローと最大流問題

最大流アルゴリズムを用いる際に重要なフローを定義する.

フロー

- 有向グラフ$\mathcal{G}(\{s\} \cup \{t\} \cup V, \varepsilon)$において,ソース$s$(入口)からシンク$t$(出口)へのモノの流れ
- 各枝eに，eの上の流れを表す実数値変数ξ_eをe上の流れを表すとすると，変数ベクトル$\boldsymbol{\xi} = (\xi_e)_{e \in \varepsilon}$をフローと定義

実行可能フロー・・・以下の二つの制約を満たす

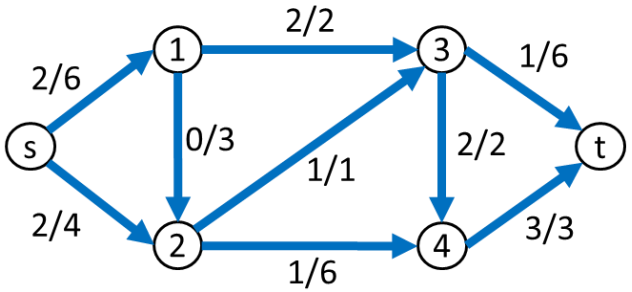
●容量制約

各枝eに，e上のフローが0以上c_e以下

●流量保存制約

s, t以外の頂点iにおいて，iから出るフローとiに入るフローの量が等しい

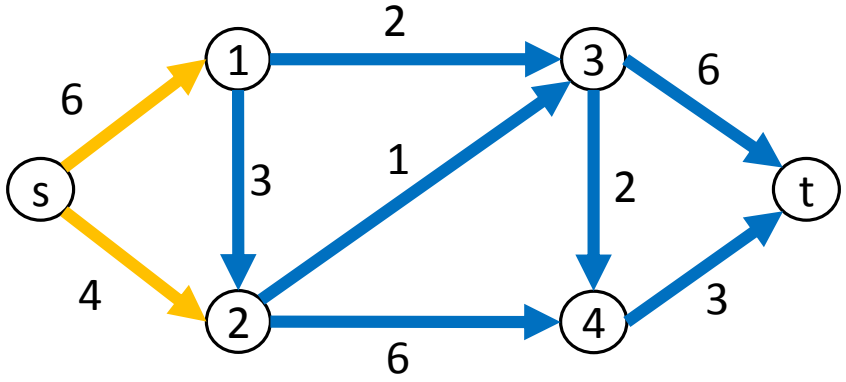
流量はsから出ていくフローξの総和



実行可能フローの例
流量は2+2=4

4.1.1.フローと最大流問題

- **最大流問題**
  流量を最大にするような実行可能フローを見つける問題
- **最大流**
  最大流問題の最適解



目的関数:

$\xi_{(s,1)} + \xi_{(s,2)} \quad \Rightarrow \quad$ 最大

制約:

$$\xi_{(1,3)} + \xi_{(1,2)} - \xi_{(s,1)} = 0, \qquad \xi_{(2,3)} + \xi_{(2,4)} - \xi_{(s,2)} - \xi_{(1,2)} = 0,$$
$$\xi_{(3,t)} + \xi_{(3,4)} - \xi_{(1,3)} - \xi_{(2,3)} = 0, \qquad \xi_{(4,t)} + \xi_{(3,4)} - \xi_{(2,4)} = 0,$$

} 流量保存

$$0 \le \xi_{(s,1)} \le 6, \quad 0 \le \xi_{(s,2)} \le 4, \quad 0 \le \xi_{(1,2)} \le 3,$$
$$0 \le \xi_{(1,3)} \le 2, \quad 0 \le \xi_{(2,3)} \le 1, \quad 0 \le \xi_{(2,4)} \le 6,$$
$$0 \le \xi_{(3,4)} \le 2, \quad 0 \le \xi_{(3,t)} \le 6, \quad 0 \le \xi_{(4,t)} \le 3,$$

} 容量制約

4.1.1.フローと最大流問題

- $s-t$カットの容量$\kappa_{s-t}(S)$はソース側の頂点からシンク側の頂点に入る枝容量の総和
- 容量制約より

  (実行可能フローξの流量) $\leq$ ($s-t$カットの容量$\kappa_{s-t}(S)$)

(実行可能フローξの流量)の最大値・・・最大流流量

($s-t$カットの容量$\kappa_{s-t}(S)$)の最小値・・・s-tカット容量の最小値

最大流最小カット定理

最大流流量＝s-tカット容量の最小値

4.1.1. フローと最大流問題

最大流流量＝6，s-tカット容量の最小値＝6

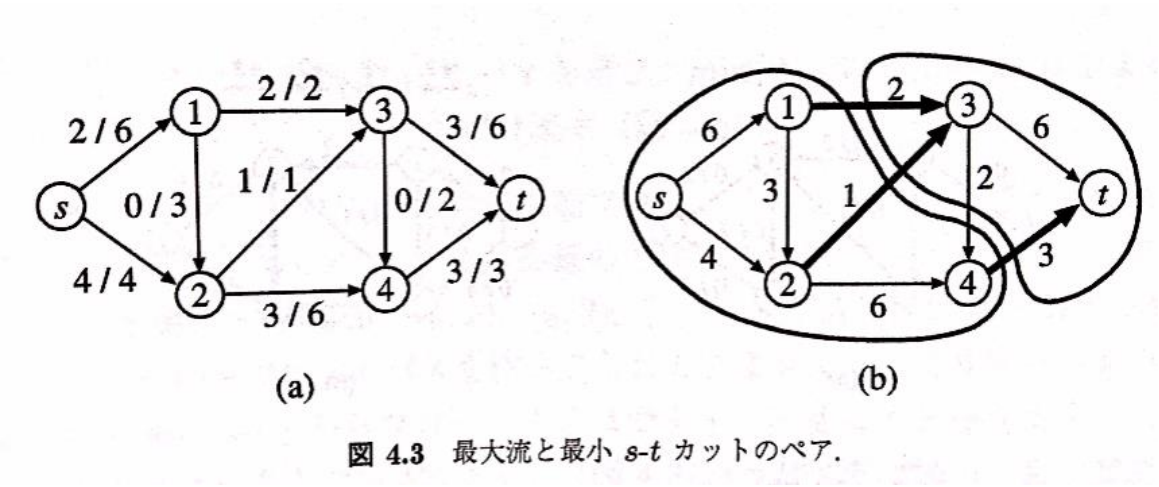


図 4.3　最大流と最小 s-t カットのペア．

アルゴリズムを用いて最大流流量を求めてみる

4.1.2.最大流アルゴリズム

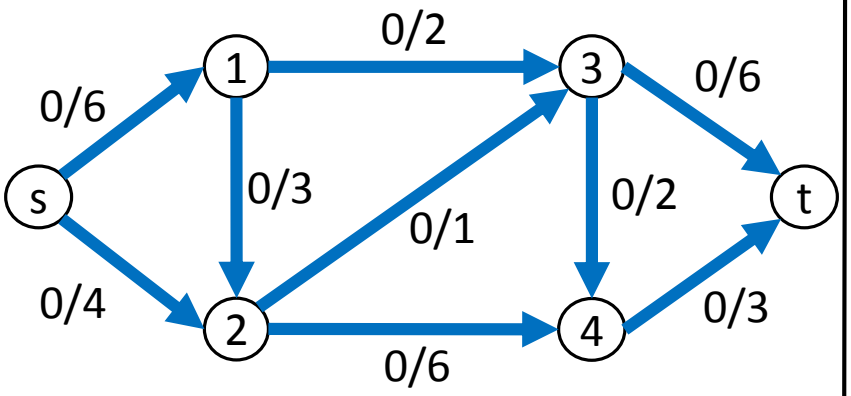
ソースsからシンクtへの有効パスをs-tパスと呼ぶ

紹介するアルゴリズム

反復法+増加パスアルゴリズム

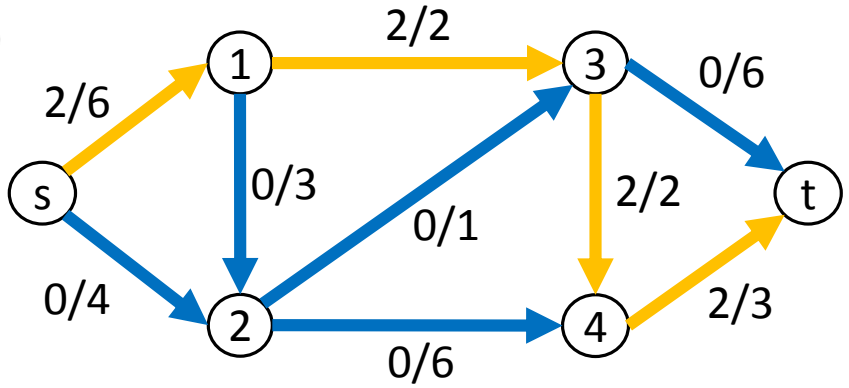
4.1.2.最大流アルゴリズム(反復法)

a)



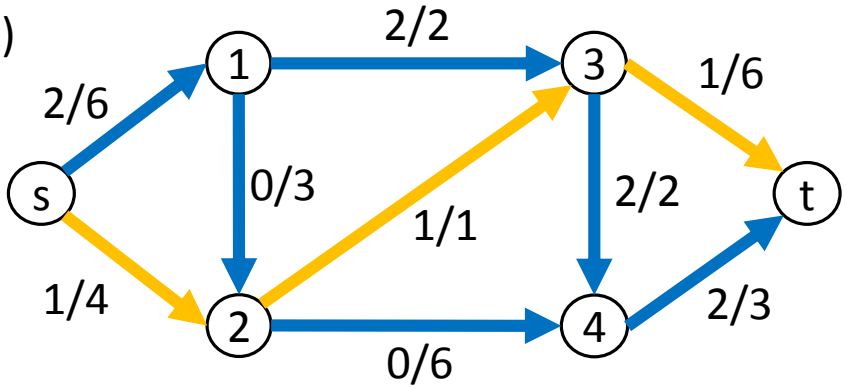
初期状態

b)



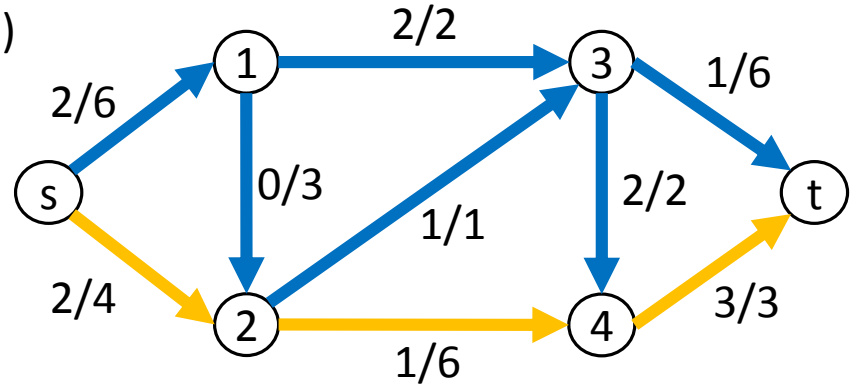
s⇒1 ⇒3 ⇒4 ⇒tというs-tパスで
$\min\{6,2,2,3\} = 2$よりフローを2流す

c)



s⇒2 ⇒3 ⇒tというs-tパスで
$\min\{4,1,6\} = 1$よりフローを1流す

d)



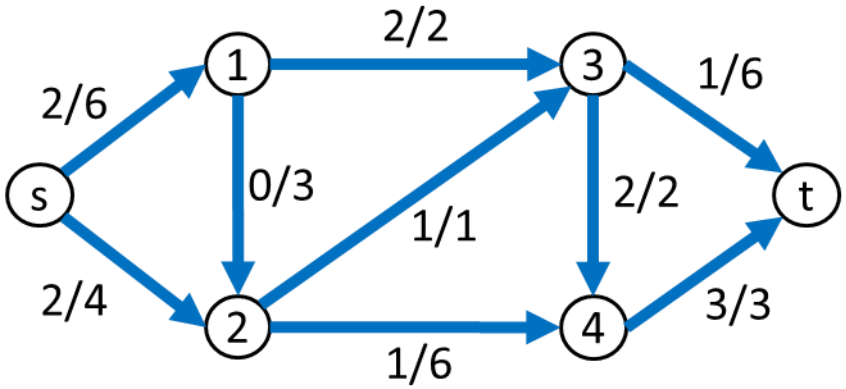
s⇒2 ⇒4 ⇒tというs-tパスで
$\min\{4-3,6,3-2\} = 1$よりフローを1流す

4.1.2.最大流アルゴリズム(反復法)

これより流量は2＋2＝4
しかし，実際の最大流は6
（P42より）

⇒単純な反復法だと最大流量を求められない

⇒「残余ネットワーク」と「フロー更新」の考え方を導入



反復法アルゴリズム

[初期化] フローを $\xi = 0$ と初期化する（図 4.4(a)）.

[反復 1] s-t パスとしてたとえば，$s \to 1 \to 3 \to 4 \to t$ を選ぶ．このパスに沿ってフローを流すことを考えると，$(s,1)$，$(1,3)$，$(3,4)$，$(4,t)$ についてはそれぞれあと 6, 2, 2, 3 だけフローを流せるので，$\min\{6, 2, 2, 3\} = 2$ より，このパスに沿ってフローを 2 だけ流す（図 4.4(b)）.

[反復 2] s-t パス $s \to 2 \to 3 \to t$ を選び，$\min\{4, 1, 6\} = 1$ より，このパスに沿ってフローを 1 だけ流す（図 4.4(c)）.

[反復 3] s-t パス $s \to 2 \to 4 \to t$ を選び，$\min\{4-1, 6, 3-2\} = 1$ より，このパスに沿ってフローを 1 だけ流す（図 4.4(d)）.

4.1.2.最大流アルゴリズム(反復法)

「残余ネットワーク」と「フロー更新」の導入

残余ネットワーク･･･容量制約を満たしつつ，ξをどの程度変化させられるかを表現する有向グラフ

$\xi_{(i,j)} = 0$を満たす$(i,j) \in \mathcal{E}$

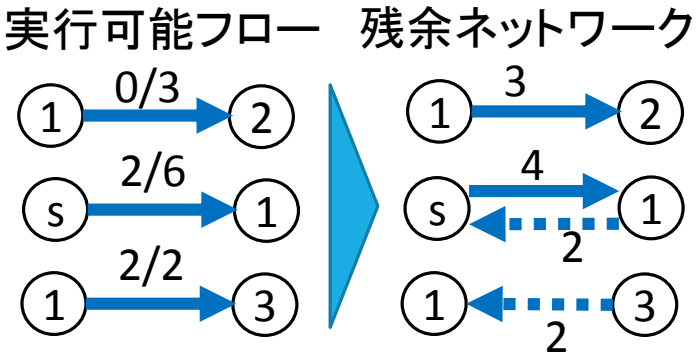
$$\Rightarrow \begin{cases} \mathcal{G}^{\xi}では(i,j)は(i,j)のまま \\ c^{\xi}_{(i,j)} = c_{(i,j)} \end{cases}$$

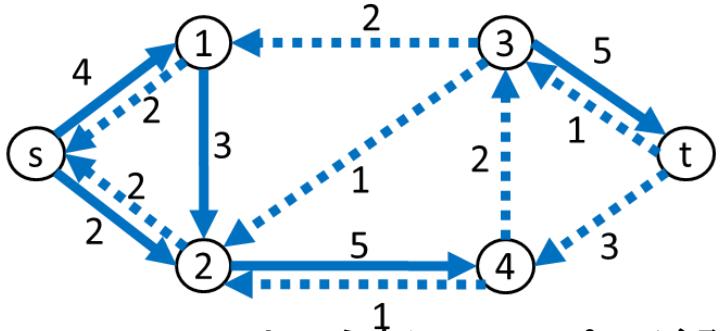
$0 \leq \xi_{(i,j)} \leq c_{(i,j)}$を満たす$(i,j) \in \mathcal{E}$

$$\Rightarrow \begin{cases} \mathcal{G}^{\xi}では(i,j)は(i,j)のまま \\ c^{\xi}_{(i,j)} = c_{(i,j)} \end{cases}$$

$\xi_{(i,j)} = 0$を満たす$(i,j) \in \mathcal{E}$

$$\Rightarrow \begin{cases} \mathcal{G}^{\xi}では(i,j)は(i,j)のまま \\ c^{\xi}_{(i,j)} = c_{(i,j)} \end{cases}$$

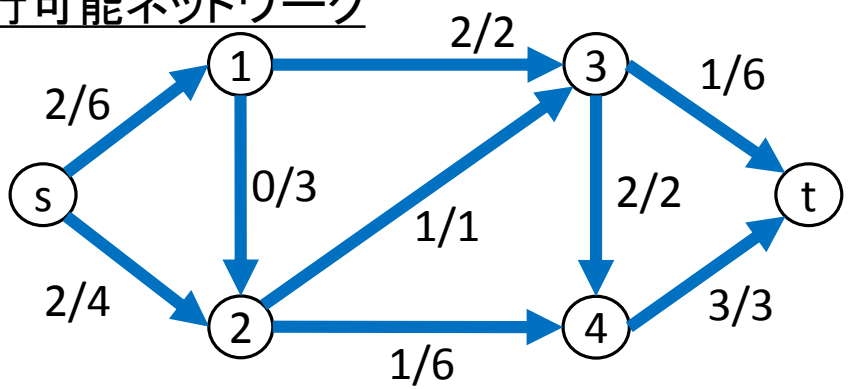




s⇒2 ⇒4 ⇒3 ⇒tという新しいs-tパスが登場

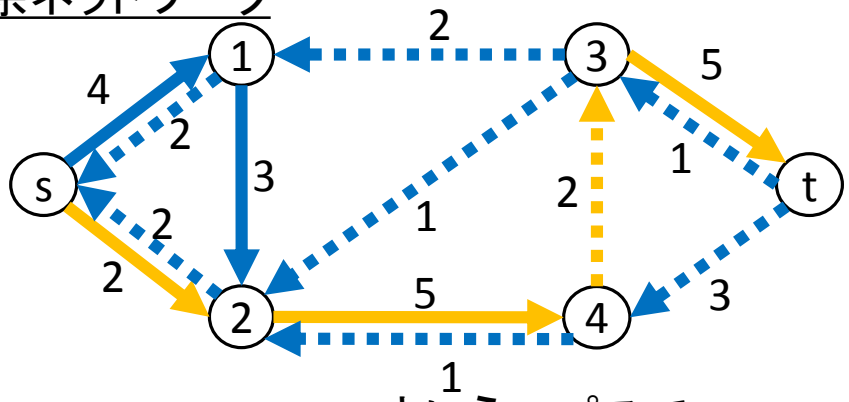
4.1.2.最大流アルゴリズム(反復法)

実行可能ネットワーク



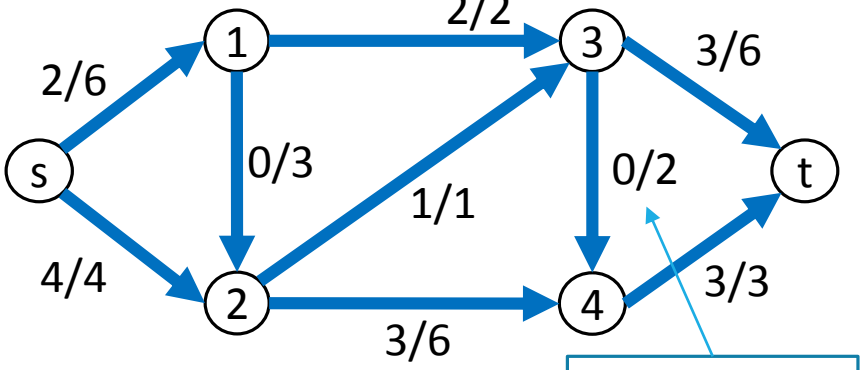
一見これ以上s-tパス
無いように見えるが

残余ネットワーク



s⇒2 ⇒4 ⇒ 3⇒tというs-tパスで
$\min\{2,5,2,5\} = 2$よりフローを2流す

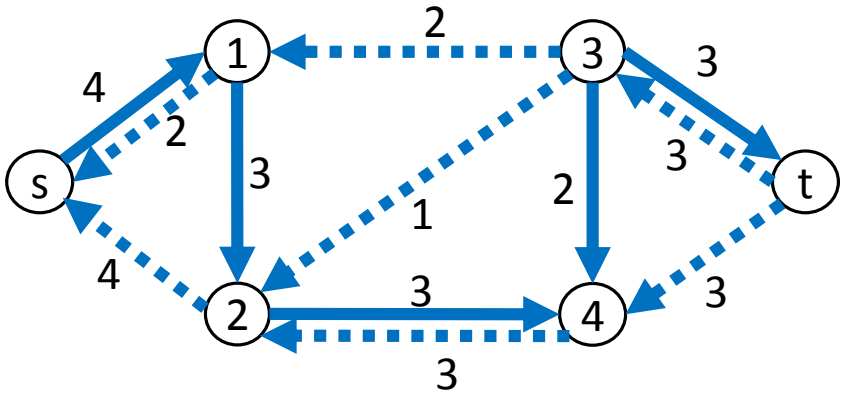
実行可能ネットワーク



流量は2＋4で6になる

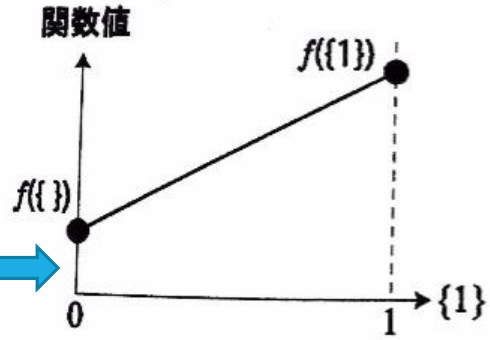
2だけ減らす

残余ネットワーク

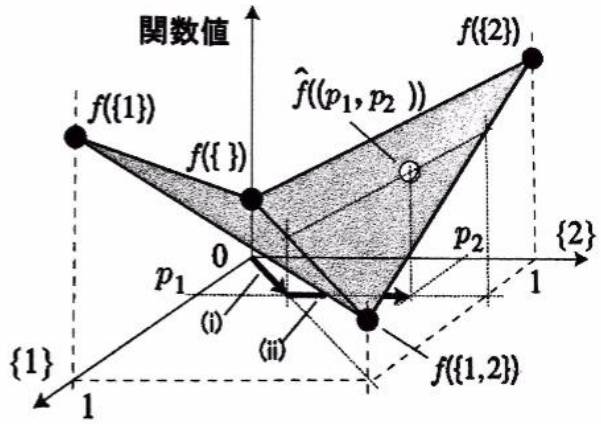


補足：劣モジュラの凸性の解説

直感的な解説.
台集合$V = \{1\}$の場合,
集合関数は
{　}or{1}の2通りの実数値への割り当て
集合関数を拡張 → $f\{\ \ \}$と$f\{1\}$間を線形補完



台集合$V = \{1,2\}$の場合
集合関数は
{　}or{1}or{2}or{1,2}の4通りの実数値
への割り当て
集合関数を拡張 →



劣モジュラの式を満たしている限り凸となる

補足：劣モジュラ関数の種類

$$f(S) + f(T) \geq f(S \cup T) + f(S \cap T) \qquad (\forall S, T \subseteq V) \qquad (1.1)$$

- 単調劣モジュラ関数
- 非負劣モジュラ関数
- 対象劣モジュラ関数
- 正規化された劣モジュラ関数

など多くの劣モジュラ関数があります．

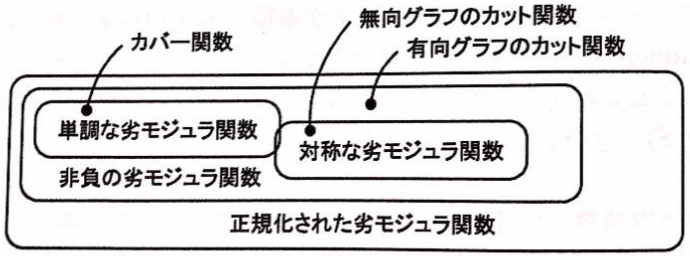


図 2.8　劣モジュラ関数のタイプ．

単調・・・より多くのものを加えたらより多くに影響を与えられる

+

劣モジュラ・・・徐々に影響力が小さくなっていく

ちなみに式(1.1)の不等号が逆になると優モジュラ関数，
等号になるとモジュラ関数と定義されます．

プログラムの紹介

ETH ZurichのAndreas Krause先生が劣モジュラ最適化のmatlabプログラムを複数公開しています.

https://las.inf.ethz.ch/sfo/

参考文献

- 河原吉伸，永野清仁　『劣モジュラ最適化と機械学習』 講談社
- 『フカシギの数え方』おねえさんといっしょ！ みんなで数えてみよう！
  https://www.youtube.com/watch?v=Q4gTV4r0zRs
- 大人になってからの再学習
  http://zellij.hatenablog.com/entry/20131004/p1
- Hakky St, 劣モジュラ最適化と機械学習
  https://www.slideshare.net/ssuser77b8c6?utm_campaign=profiletracking&utm_medium=sssite&utm_source=ssslideview
- アルゴリズムが世界を変えるSeason3
  https://www.youtube.com/watch?v=Z7eMzSHGGAE&t=339s
- Matlab Toolbox for Submodular Function Optimization (v 2.0), Andreas Krause

https://las.inf.ethz.ch/sfo/

- 日光観光関連（山楽，ホームメイト，MAP&NEWS）