離散最適化基礎論　第 1 回
アルゴリズム的問題解決と計算複雑性

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2019 年 10 月 8 日

最終更新：2019 年 10 月 16 日　07:34

概要

主題

離散最適化のトピックの 1 つとして離散最適化における計算困難性を取り上げ，計算困難性の考え方が離散最適化にどう応用されるのか，理解する

なぜ講義で取り扱う？

- 重要な概念であるから
- アルゴリズム設計の根幹を成すから
- まとまって講義として取り扱われることが，あまりないから

スケジュール 前半 (予定)

- ⋆ 休み (大学院入学式)　(10/1)
- 1 アルゴリズム的問題解決と計算複雑性　(10/8)
- 2 速習 P vs NP 問題　(10/15)
- ⋆ 休み (祝日)　(10/22)
- 3 充足可能性問題とその変種　(10/29)
- 4 グラフに関する問題 (1)：部分集合の選択　(11/5)
- 5 グラフに関する問題 (2)：経路の選択　(11/12)
- 6 集合族に関する問題 (1)：グラフとの関連　(11/19)
- 7 集合族に関する問題 (2)：発展　(11/26)

注意：予定の変更もありうる

スケジュール 後半 (予定)

- 8 数値が関わる問題 (1)：2 分割問題　(12/3)
- ⋆ 出張のため休講？　(12/10)
- 9 数値が関わる問題 (2)：3 分割問題　(12/17)
- 10 冬期休業　(12/24, 31)
- 11 平面性が関わる問題　(1/7)
- 12 計算幾何学に関する問題　(1/14)
- 13 文字列に関する問題　(1/21)
- 14 アルゴリズム的問題解決：再考　(1/28)
- 15 予備　(2/4)
- ⋆ 祝日のため休み　(2/11)

注意：予定の変更もありうる

情報

教員

- 岡本 吉央 (おかもと よしお)
- 居室：西 4 号館 2 階 206 号室
- E-mail：okamotoy@uec.ac.jp
- Web：http://dopal.cs.uec.ac.jp/okamotoy/

講義資料

- Web：http://dopal.cs.uec.ac.jp/okamotoy/lect/2019/npc/
- 注意：資料の入手は各学生が自ら行う
- 講義前日の夕方 18 時までに，ここに置かれる
- Twitter (@okamoto7yoshio)：置かれたことを知らせる tweet

講義資料

http://dopal.cs.uec.ac.jp/okamotoy/lect/2019/npc/

- スライド
- 印刷用スライド：8 枚のスライドを 1 ページに収めたもの

「印刷用スライド」は各自印刷して持参すると便利

授業の進め方

講義 (85 分)

- スライドと板書で進める
- スライドのコピーに重要事項のメモを取る

退室 (5 分) ←重要

- コメント (授業の感想，質問など) を紙に書いて提出する (匿名可)
- コメントとそれに対する回答は (匿名で) 講義ページに掲載される

オフィスアワー：アポイントメントによる

- 質問など

評価

2 回のレポートのみによる

- レポート 1 (50 点満点)
  - 要項説明：12 月 3 日 (火)
  - 提出締切：12 月 25 日 (水)
- レポート 2 (50 点満点)
  - 要項説明：1 月 28 日 (火)
  - 提出締切：2 月 12 日 (水)

要項説明 は 講義 web ページ でも行う

教科書・参考書

教科書
- 指定しない

全般的な参考書
- M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Co., 1979.
- B. Korte, J. Vygen, Combinatorial Optimization, 6th Edition, Springer, 2018.

その他，研究論文

この講義の約束

- 私語は慎む
- 携帯電話等はマナーモードにする
- この講義と関係のないことを (主に電子機器で) しない
- 音を立てて睡眠しない

約束が守られない場合は退席してもらう場合あり

目次

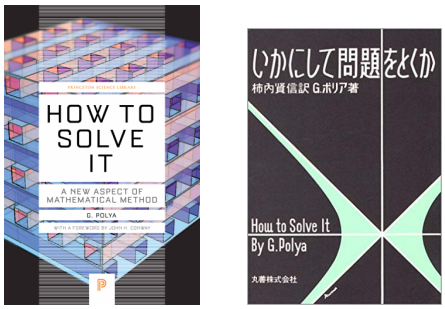
ジョージ・ポリア

ジョージ・ポリア (ジェルジ・ポーヤ，György Pólya) 1887–1985

ハンガリー生まれの数学者

http://www-history.mcs.st-andrews.ac.uk/Biographies/Polya.html

『いかにして問題を解くか』



1945 年出版，数学的問題解決に関する手引書

問題解決の流れ

問題解決の流れ (ポリアによる)

(1) 問題を理解すること	(understand)
(2) 計画をたてること	(plan)
(3) 計画を実行すること	(carry out)
(4) ふり返ってみること	(look back)

ポリアの射程： 数学的問題解決
本講義の射程： アルゴリズム的問題解決

目標

上の「問題解決の流れ」を「アルゴリズム的問題解決」で行う

(1) 問題を理解すること

アルゴリズム的問題解決における「問題を理解すること」

アルゴリズムがすべきことを定めること (直感的な言い方)

入力 ：アルゴリズムが何を受け取るのか
出力 ：アルゴリズムが何をもたらすのか
条件 ：出力が満たすべき性質は何なのか
(評価)：出力の良さをどのように測るのか

注釈 1

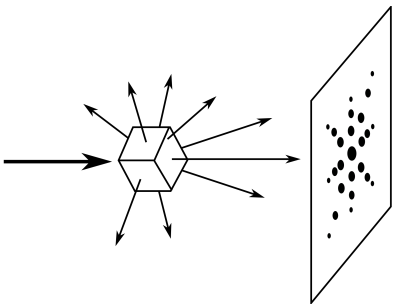
分野によっては，次のような用語を使うことがある
条件 = 制約　　評価 = 目的，目標

注釈 2

「出力の評価」と「アルゴリズムの評価」は別概念なので，区別する

例：X 線回折による結晶構造解析 (1)

測定全体にかかる時間を減らしたい　　Bland, Shallcross ('89)



ビフェニルを 1.00Å の波長で測定するとき，14464 個の配置を使う

https://commons.wikimedia.org/wiki/File:Cliche_de_laue_principe.svg

例：X 線回折による結晶構造解析 (2)

状況を詳しく見てみると…

- 14464 個の配置はどの順で測定しても良い
- 測定 1 回にかかる時間は (無視できるほど) 短い
- 配置を変更する時間が (無視できないほど) 長い

また，他の物質，他の波長にも対応できるようにしたい

ここから，次のように問題を定める

解くべき問題の定式化

- 入力：X 線回折装置の配置の集合，配置間の変更時間
- 出力：配置の変更順
- 条件：特になし
- 評価：配置の総変更時間を最小化

「問題」とは？

厄介な側面

日本語の「問題」が表す概念は，2 通りある

- Problem (問題)
  - アルゴリズムを設計する対象
- Instance (問題例)
  - 解くべき個々の問題
  - 例：「ビフェニル，波長 1.00Å，配置 14464 個」といった特定の状況

問題解決の流れ (再)

問題解決の流れ (ポリアによる)

(1)	問題を理解すること	(understand)
(2)	計画をたてること	(plan)
(3)	計画を実行すること	(carry out)
(4)	ふり返ってみること	(look back)

ポリアの射程： 数学的問題解決
本講義の射程： アルゴリズム的問題解決

目標

上の「問題解決の流れ」を「アルゴリズム的問題解決」で行う

(2) 計画をたてること

アルゴリズム的問題解決における「計画をたてること」

1. アルゴリズム設計の方針を立てること
2. アルゴリズムを設計すること

アルゴリズム設計の方針 とは？

- 我々が目標とする「実行時間」と「メモリ消費量」の特定
- 我々が達成できる「実行時間」と「メモリ消費量」の特定
  ← この講義の内容

アルゴリズム設計の方針

現実問題の付帯条件を細かく検討する必要がある

スケジューリングのような問題にも様々

- 中学校の時間割作成
  教員数 30 名程度，学級数 15 程度，1 年に 1 回作成
  計算にかけられる時間：1 日程度 ← 目標とする実行時間
- 銀行 ATM の現金補充計画作成
  ATM 数 200 個程度，拠点数 20 個程度，1 年に 1 回程度作成
  計算にかけられる時間：1 週間程度 ← 目標とする実行時間
- 看護師の勤務表作成
  看護師数 25 名程度，3 交替制，1ヵ月に 1 回作成
  計算にかけられる時間：1 時間程度 ← 目標とする実行時間

「我々の目標とするもの」と「我々の達成できるもの」のギャップが重要

- 目標が達成できる ⇒ 達成すればよい
- 目標が達成できない ⇒ どうすれば？ (第 14 回で扱う予定)

問題解決の流れ (再)

問題解決の流れ (ポリアによる)

(1)	問題を理解すること	(understand)
(2)	計画をたてること	(plan)
(3)	計画を実行すること	(carry out)
(4)	ふり返ってみること	(look back)

ポリアの射程： 数学的問題解決
本講義の射程： アルゴリズム的問題解決

目標

上の「問題解決の流れ」を「アルゴリズム的問題解決」で行う

(3) 計画を実行すること

アルゴリズム的問題解決における「計画を実行すること」

- アルゴリズムをプログラムとして実現 (実装) すること
- プログラムを実行すること

注意

「アルゴリズム」と「プログラム」は明確に区別される

アルゴリズム	手続きの抽象的記述	(主に数学による)
プログラム	手続きの具体的記述	(主にプログラミング言語による)

つまり，

- 「速いアルゴリズム」と「速いプログラム」は区別しないといけない
- 「アルゴリズムがプログラムとして実現できるか」は分からない

(4) ふり返ってみること

アルゴリズム的問題解決における「ふり返ってみること」

- 問題が正しく理解できているか，評価する
- アルゴリズムとプログラムに満足できるか，評価する

評価ができたら，また (1) に戻って，繰り返す

- 解くべき問題が理解できてないと 判明することも多い

目次

問題

問題 (problem) とは？ (直感的定義)

問題とは，次を定めることで記述される

- 入力 ：アルゴリズムが何を受け取るのか
- 出力 ：アルゴリズムが何をもたらすのか
- 条件 ：出力が満たすべき性質は何なのか
- (評価)：出力の良さをどのように測るのか

例：整列問題

- 入力 ：整数列 A
- 出力 ：A を並び替えた整数列
- 条件 ：A の要素は昇順に並んでいる

問題例

問題例 (instance) とは？

問題 P の問題例とは，P の入力 1 つのこと

例：整列問題

次の 1 つ 1 つは整列問題の問題例

- $(1, 5, 3, 8, 2)$
- $(5, 8, 1, 4, 6, 2, 0, 3, 7)$

注意

日常用語では，instance を問題と呼ぶことが (頻繁に) ある

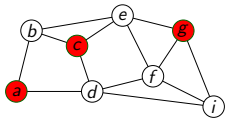
つまり，意識的に区別するよう，注意しないといけない

別の例

別の例：最大独立集合問題

- 入力：無向グラフ G
- 出力：G の独立集合 S
- 評価：S の要素数の最大化

無向グラフ G の独立集合とは，G の頂点部分集合 S で
S のどの 2 頂点間にも辺が存在しないもの



このグラフは 最大独立集合問題 の問題例

目次

アルゴリズムの複雑性

アルゴリズムの複雑性

問題 P を解くアルゴリズム A の複雑性といったら，
A の実行にかかる計算資源の量のこと

ここでは，「計算資源 $=$ 実行時間」とする

慣習

興味があるのは「最悪時実行時間の漸近的振舞い」

$\rightsquigarrow$ オーダー記法

アルゴリズムの複雑性：オーダー記法 (1)

問題 P を解くアルゴリズム A，非減少関数 $f\colon \mathbb{N} \to \mathbb{N}$

アルゴリズムの複雑性に関する記法：上界 (upper bound)

A の最悪時実行時間が $O(f(n))$ であるとは，

ある実数 C が存在して，P の任意の入力 I に対して，

$$I \text{ を入力したときの } A \text{ の実行時間 } \leq C \cdot f(|I|)$$

が成り立つこと

ただし，$|I|$ は入力 I のサイズ

ここで，

- I のサイズの測り方は場合による
- A の実行時間の測り方は場合による

「場合による」ので，しっかりと定めないといけない

アルゴリズムの複雑性：オーダー記法 (2)

問題 P を解くアルゴリズム A，非減少関数 $f\colon \mathbb{N} \to \mathbb{N}$

アルゴリズムの複雑性に関する記法：下界 (かかい，lower bound)

A の最悪時実行時間が $\Omega(f(n))$ であるとは，

ある実数 C が存在して，任意の $n \in \mathbb{N}$ に対して，
$|I| = n$ であるような P のある入力 I が存在して，

$$I \text{ を入力したときの } A \text{ の実行時間 } \geq C \cdot f(|I|)$$

が成り立つこと

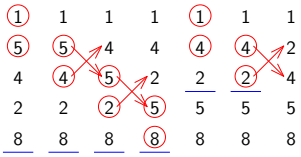
ここで，

- I のサイズの測り方は場合による
- A の実行時間の測り方は場合による

「場合による」ので，しっかりと定めないといけない

例：整列問題を解くバブルソート

入力：$(1, 5, 4, 2, 8)$



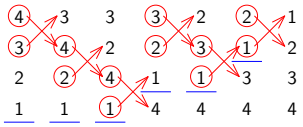
「実行時間 $=$ 比較回数」,「入力のサイズ $=$ 入力の要素数」とすると，

バブルソートの実行時間 $= O(n^2)$

例：整列問題を解くバブルソート — 下界

任意の $n \in \mathbb{N}$ に対して，次のような入力 I_n を考える

$$I_n = (n, n-1, n-2, \ldots, 3, 2, 1)$$



「実行時間 $=$ 比較回数」,「入力のサイズ $=$ 入力の要素数」とすると，

バブルソートの実行時間 $= \Omega(n^2)$

問題の複雑性

問題の複雑性

問題 P の複雑性といったら，

問題 P を解く「最もよい」アルゴリズム A の実行にかかる計算資源の量のこと

ここでは，「計算資源 $=$ 実行時間」とする

注意

「問題の複雑性」と「アルゴリズムの複雑性」は違う概念

問題の複雑性：オーダー記法 (1)

問題 P，非減少関数 $f : \mathbb{N} \to \mathbb{N}$

問題の複雑性に関する記法：上界 (upper bound)

P の最悪時複雑性が $O(f(n))$ であるとは，

P を解くあるアルゴリズム A と ある実数 C が存在して，
P の任意の入力 I に対して，

$$I \text{ を入力したときの } A \text{ の実行時間 } \leq C \cdot f(|I|)$$

が成り立つこと

ただし，$|I|$ は入力 I のサイズ

問題の複雑性：オーダー記法 (2)

問題 P，非減少関数 $f : \mathbb{N} \to \mathbb{N}$

問題の複雑性に関する記法：下界 (かかい，lower bound)

P の最悪時複雑性が $\Omega(f(n))$ であるとは，

ある実数 C が存在して，
P を解く任意のアルゴリズム A と 任意の $n \in \mathbb{N}$ に対して，
$|I| = n$ であるような P のある入力 I が存在して，

$$I \text{ を入力したときの } A \text{ の実行時間 } \geq C \cdot f(|I|)$$

が成り立つこと

例：整列問題

事実

- 整列問題の最悪時複雑性は $O(n \log n)$
  - 最悪時実行時間が $O(n \log n)$ である整列アルゴリズムが存在
- 整列問題の最悪時複雑性は $\Omega(n \log n)$
  - どんな整列アルゴリズムの最悪時実行時間も $\Omega(n \log n)$

$O(f(n))$ かつ $\Omega(f(n))$ であることを $\Theta(f(n))$ と書くことがあるので，

- 整列問題の最悪時複雑性は $\Theta(n \log n)$

注意

整列問題のように最悪時実行時間の上界と下界 (のオーダー) として一致するものが知られている問題は数少ない

例：最小全域木問題 (minimum spanning tree problem)

頂点数 n，辺数 m とする

最小全域木問題の最悪時複雑性 (上界)

- $O(m \log n)$ (Kruskal 法)
- $O(m + n \log n)$ (Prim 法)
- $O(m\,\alpha(m, n))$ (Chazelle '00 のアルゴリズム)

$\alpha(m, n) =$ アッカーマン逆関数 (とてもゆっくり増加する関数)

最小全域木問題の最悪時複雑性 (下界)

- $\Omega(m + n)$ (入力を読むのに必要)

目次

判定問題

判定問題 (決定問題，decision problem) とは？

出力が「Yes」か「No」であるような問題

判定問題 の例

合成数判定問題

- 入力：正整数 $n \geq 2$
- 出力：Yes か No
- 条件：n が合成数ならば Yes，n が素数ならば No

- 入力が 6 → 出力は Yes
- 入力が 7 → 出力は No

判定問題ではない問題 の例

整列問題

最適化問題

最適化問題 (optimization problem) とは？

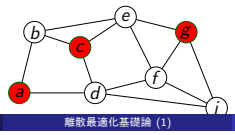
評価が次のいずれかである問題

- 出力の○○を最小とすること (最小化)
- 出力の○○を最大とすること (最大化)

最適化問題の例

最大独立集合問題

- 入力：無向グラフ G
- 出力：G の独立集合 S
- 評価：S の要素数の最大化



最適化問題と判定問題

重要な事実

最適化問題を判定問題に帰着できる場合がある

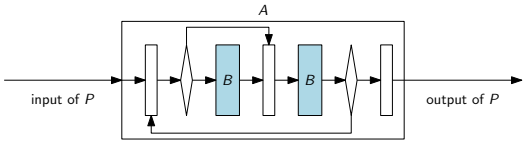
「帰着」はこの講義のキーワードの 1 つ

帰着とは？ (直感的な定義)

問題 P を問題 Q に帰着する (還元する，reduce) とは，

問題 Q を解くアルゴリズム B をサブルーチンとして
問題 P を解くアルゴリズム A を構成すること

帰着にも いろいろな種類のものがある (第 2 回講義)



帰着の例：最大独立集合問題 (1)

最適化問題 P

最大独立集合問題

- 入力：無向グラフ G
- 出力：G の独立集合 S
- 評価：S の要素数の最大化

判定問題 Q

独立集合問題

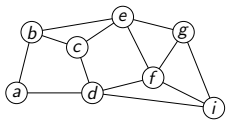
- 入力：無向グラフ G，自然数 $k \in \mathbb{N}$
- 出力：Yes か No
- 条件：G が要素数 k 以上の独立集合を持つ ⇒ Yes
  G が要素数 k 以上の独立集合を持たない ⇒ No

独立集合問題を解くアルゴリズム B が存在する と仮定する

帰着の例：最大独立集合問題 (2)

最大独立集合問題を解くアルゴリズム

1. For $i = 1, \ldots, G$ の頂点数:
2. $\quad B$ に G と i を入力
3. $\quad B$ の出力が No ならば，最適値 $i - 1$ を出力して，停止
4. $\quad B$ の出力が Yes ならば，何もしない



- $i = 1 \to$ Yes
- $i = 2 \to$ Yes
- $i = 3 \to$ Yes
- $i = 4 \to$ No

$\therefore$ 最適値 $= 3$

不満

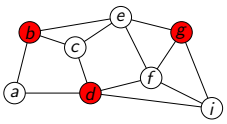
「最適値」は分かるが,「最適解」が分からない

帰着の例：最大独立集合問題 (3)

最大独立集合問題を解くアルゴリズム (続き)

$V = G$ の頂点集合，$k =$ 最適値 (前ページで計算済み)

5. For $v \in V$:
6. $\quad B$ に $G - v$ と k を入力
7. $\quad B$ の出力が Yes ならば，
   1. G における要素数 k の独立集合で，v を含まないものが存在
   2. G から v を除去する
8. $\quad B$ の出力が No ならば，
   1. G における要素数 k の独立集合は，どれも v を含む
   2. v を最適解に含める
   3. G から v と v の隣接頂点を除去し，k を $k - 1$ で置き換える



$k = 1$　最適解 $= \{b, d, g\}$

- $v = a \to$ Yes
- $v = b \to$ No
- $v = d \to$ No
- $v = g \to$ No

目次

今日のまとめ

アルゴリズム的問題解決

(1) 問題を理解すること
(2) 計画をたてること
(3) 計画を実行すること
(4) ふり返ってみること

異なる概念の認識

- 問題と問題例
- アルゴリズム設計とアルゴリズム設計方針
- アルゴリズムとプログラム
- 最適化問題と判定問題
  - 最適化問題は判定問題に帰着できる (場合がある)

次回の予告

次回の内容

- 次の基本概念を理解する
  - P, NP, NP 困難，NP 完全，証拠

文献案内

- G. Polya, How to Solve It, Princeton University Press, 1945.
- R.G. Bland, D.F. Shallcross, Large travelling salesman problems arising from experiments in X-ray crystallography: A preliminary report. Operations Research Letters 8 (1989) pp. 125–128.
- B. Chazelle, A minimum spanning tree algorithm with inverse-Ackermann type complexity. Journal of the ACM 47 (2000) pp. 1028–1047.