粒子群最適化
Particle Swarm Optimization (PSO)

- **最適化とは**
- **最適化の難しさ**
- **最適化手法の種類**
- **PSOの概要**
- **PSOの手順**

最適化問題とは

変数や目的関数，制約条件などの要素を定義した上で

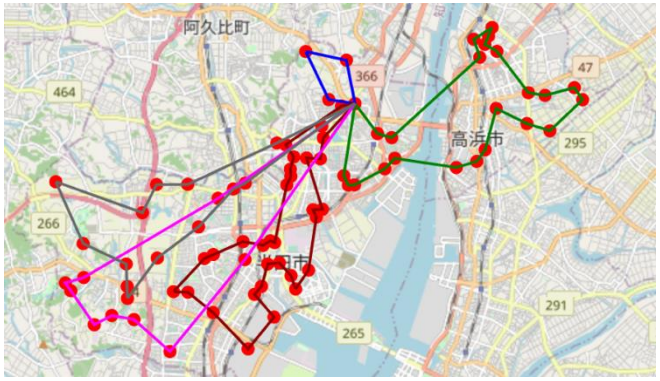
目的関数の値を最大化/最小化する**解**を求める問題

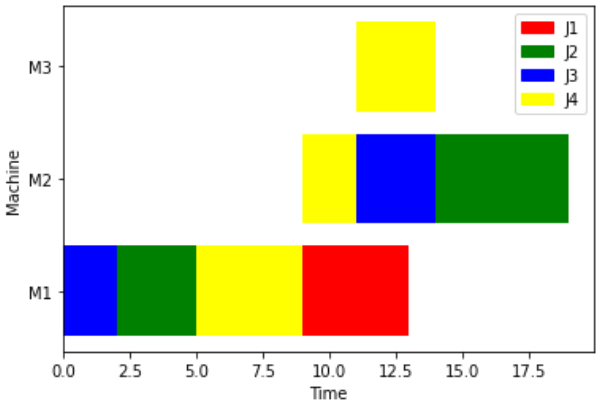
身の回りの最適化問題

- シフトスケジュールの作成
- 時間割の作成
- 配送，送迎の経路
- タスクの割り当て

膨大な組合せの中から
最も良い解を探す問題

最適解





熟練度	名前	1	2	3	4	5	6	7
ベテラン	n1	日		日	準		日	日
ベテラン	n2	準		日	日	準		
中堅	n3	日	日	準	準		日	
中堅	n4	深	準	深		深	準	日
新人	n5		日		深	準		準
新人	n6	準	深	準		日	深	
新人	n7				日			深
新人	n8	深	深	準		日	深	

最適化における目的

YAHOO! JAPAN 路線情報 https://transit.yahoo.co.jp/ より引用

岐阜→博多

ほかに候補があります　出発地：岐阜駅　到着地：博多駅　再検索

早 到着時刻順	楽 乗換回数順	安 料金の安い順	
ルート1	3時間10分	35,550円 乗換：3回	早
ルート2	3時間19分	34,770円 乗換：3回	
ルート3	3時間20分	34,820円 乗換：2回	

早 到着時刻順	楽 乗換回数順	安 料金の安い順	
ルート1	⚠6時間19分	18,160円 乗換：1回	楽
ルート2	12時間25分	14,480円 乗換：1回	楽 安
ルート3	3時間20分	34,820円 乗換：2回	

早 到着時刻順	楽 乗換回数順	安 料金の安い順	
ルート1	12時間25分	14,480円 乗換：1回	楽 安
ルート2	⚠6時間19分	18,160円 乗換：1回	楽
ルート3	⚠6時間51分	20,420円 乗換：2回	

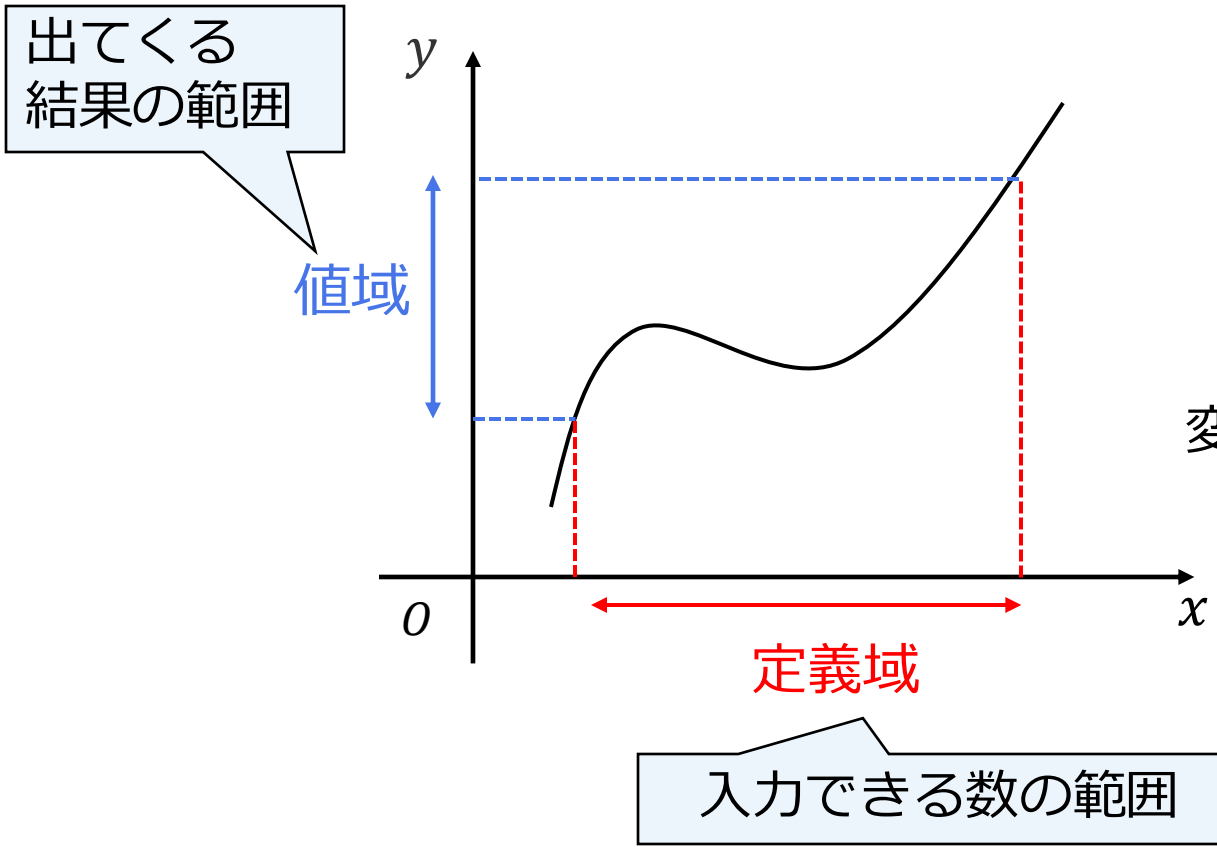
同じ出発地と目的地でも，利用者によって「何を最適化したいか」は異なる

- **到着時刻を最優先**する人
  → 最短時間ルートを選択
- **乗り換え回数を減らしたい**人
  → 乗換が少ないルートを選択
- **料金を安くしたい**人
  → 最安ルートを選択

全てを同時に最適化できるプランは…？

関数とは

- 「入力と出力を結びつけるルール全般」を**写像**という
- その中でも，**数と数を結びつける特別な写像**を**関数**という
- 「入力1つにつき出力1つ」＝**関数のルール**



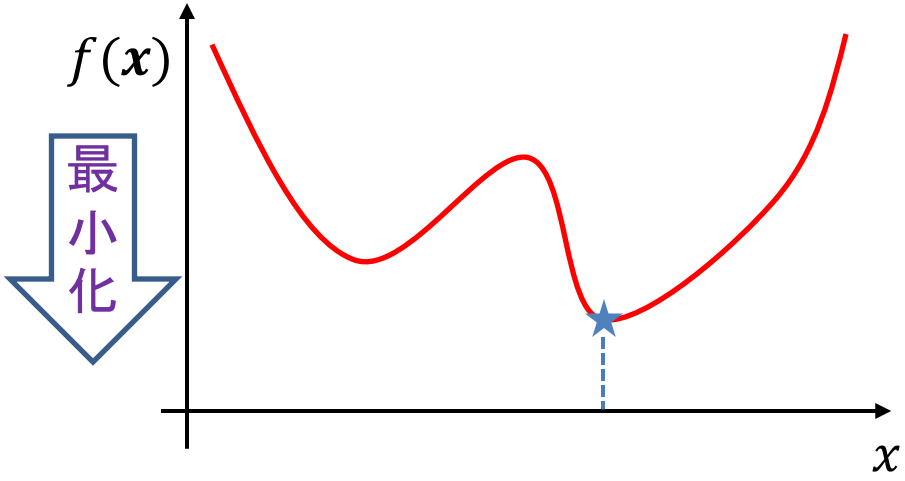
目的関数の最大化/最小化

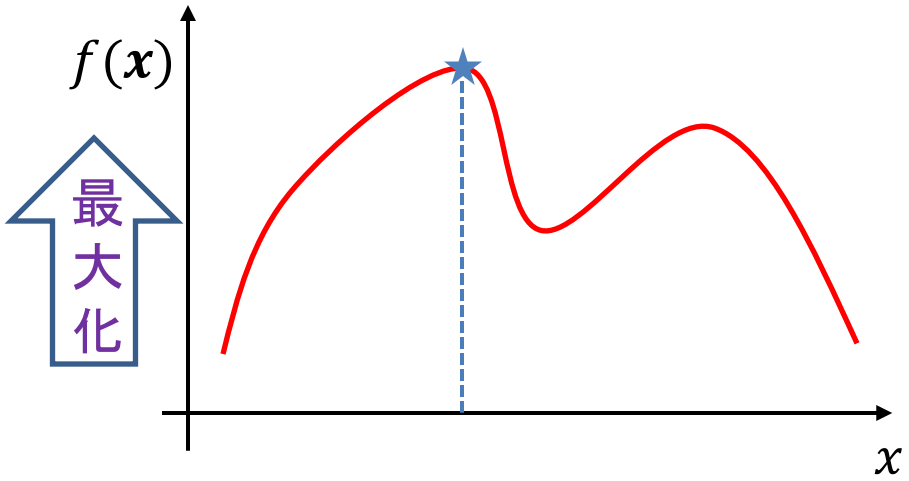
最適化問題を解くということ:

=> 目的関数 $f(x)$の値を最大化/最小化する解 x を求める

事前に決められる

増減表書けば(微分で)わかる？





- 目的関数 $f(x)$ は，ある解 x がどれほど良いか，または悪いかを数値的に評価するための関数
- ある解 x は，例えば，
  - 乗り換え問題：利用する列車やバスの組み合わせと順序
  - 設計問題: 構造物を構成する部品の大きさや重さ

最適化問題の例：遠足におけるおやつの選び方

決まりごと：
- おやつは250円まで
- リストから1つずつ選択可
- ジュースはダメ
- バナナはダメ

持っていく
お菓子を選ぶ

選べるおやつのリスト

値段：100円
価値：50

値段：70円
価値：30

値段：90円
価値：30

値段：60円
価値：40

値段：30円
価値：20

制約条件

決まりごとを守ったうえで，
最も価値の合計が高い組合せは？

＝ 組合せ最適化問題

考え方

- 5種類のお菓子の情報を表にまとめる

	チョコ	ポッキー	ドーナッツ	ポテト	アメ
値段	100	70	90	60	30
価値	50	30	30	40	20

- 答えの表し方

	チョコ	ポッキー	ドーナッツ	ポテト	アメ
解の例	選ぶ	選ばない	選ぶ	選ばない	選ぶ

値段の合計：220，　価値の合計：100

答えの表現方法

チョコ	ポッキー	ドーナッツ	ポテト	アメ
選ぶ	選ばない	選ぶ	選ばない	選ぶ

	チョコ	ポッキー	ドーナッツ	ポテト	アメ
ある解 $\boldsymbol{x}$	1	0	1	0	1

- 選ぶ: 1
- 選ばない: 0

離散値で表現

符号化：答えを「0と1で表現」（2進数）

- コンピュータの内部では，データを2進数で表現する
- 2進数の1桁が1ビット
- 解 $\boldsymbol{x}$ を5ビットで表現できる

ベクトルなのでボールド（太字）

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_5)$$

00000
00001
00010
00011
⋮

何通りの組合せ？

$$2^5 = 32$$

ある解の値段と価値の合計

ある解 $x = (1, 0, 1, 0, 1)$

	チョコ	ポッキー	ドーナッツ	ポテト	アメ
値段	100	70	90	60	30
解 x	1	0	1	0	1
	100	＋	90	＋	30

値段の合計＝220

	チョコ	ポッキー	ドーナッツ	ポテト	アメ
価値	50	30	30	40	20
解 x	1	0	1	0	1
	50		30		20

価値の合計＝100

これは最適な答え（最適解）？
最適解を発見するためには？

->全ての組合せ（32通り）を調べてみる

Brute-force search（**力まかせ探索**）

おやつ選択の目的関数

- おやつ選択の目的は，
  「できるだけ価値が高くなるようにおやつを選ぶこと」
- この「価値の合計」を計算する式が目的関数

0 or 1 を
決めること

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} v_i x_i$$

$n = 5$

解 $\boldsymbol{x} = (1, 0, 1, 1, 0)$

価値 $\boldsymbol{v} = (50, 30, 30, 40, 20)$

値段の制約がなければ・・・　すべて選べば価値は最大

価値の合計を最大化したいが，**値段の合計は上限(250円)以内**
にしなければならない (**制約条件**を満たす必要がある)

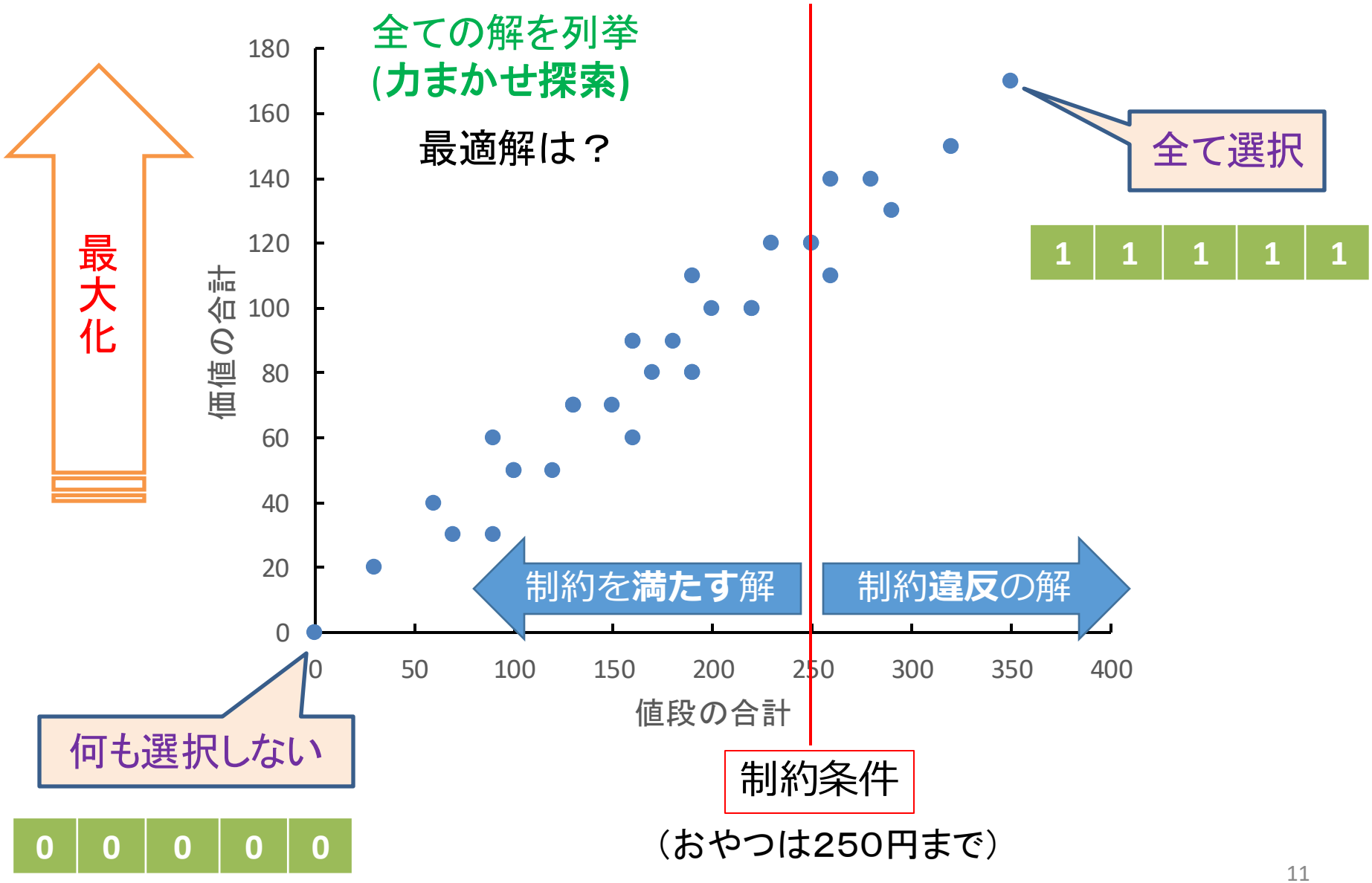
$$\sum_{i=1}^{n} w_i x_i \leq W$$

値段 $\boldsymbol{w} = (100, 70, 90, 60, 30)$

予算 $W = 250$

=> **制約あり**なら，選び方に工夫が必要（**問題の難しさ**）

おやつ選択問題の32個の答えの分布



探索する範囲（解空間）

5個なら32通り．では，50個なら何通り？

$$2^5 = 32$$

$$2^{10} = 1024$$

$$\vdots$$

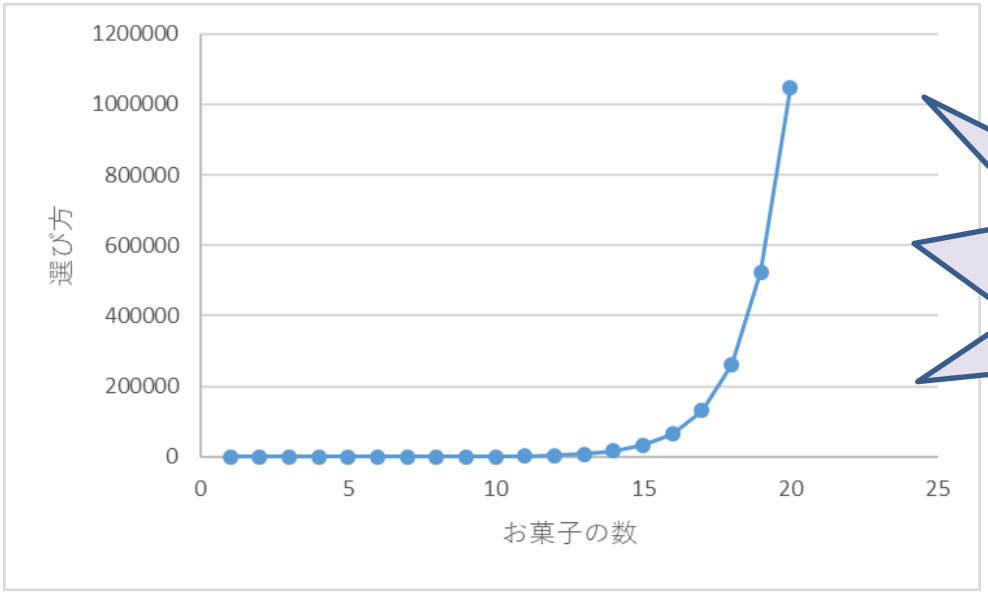
$$2^{50} = \ ?$$

$y = 2^x$ のグラフは？

https://www.geogebra.org/

最適化問題の難しさ

組合せが多すぎて，全部試すのは無理！



組合せ爆発

- **課題**
  - 「全部試さないで，いい答えをどうやって探すか？」

- **解決策**
  - **最適化アルゴリズム**（賢い探索方法）を使う
  - 良さそうな解候補を**少しずつ改良**して，最適解に近づく

最適化アルゴリズムの2つのタイプ

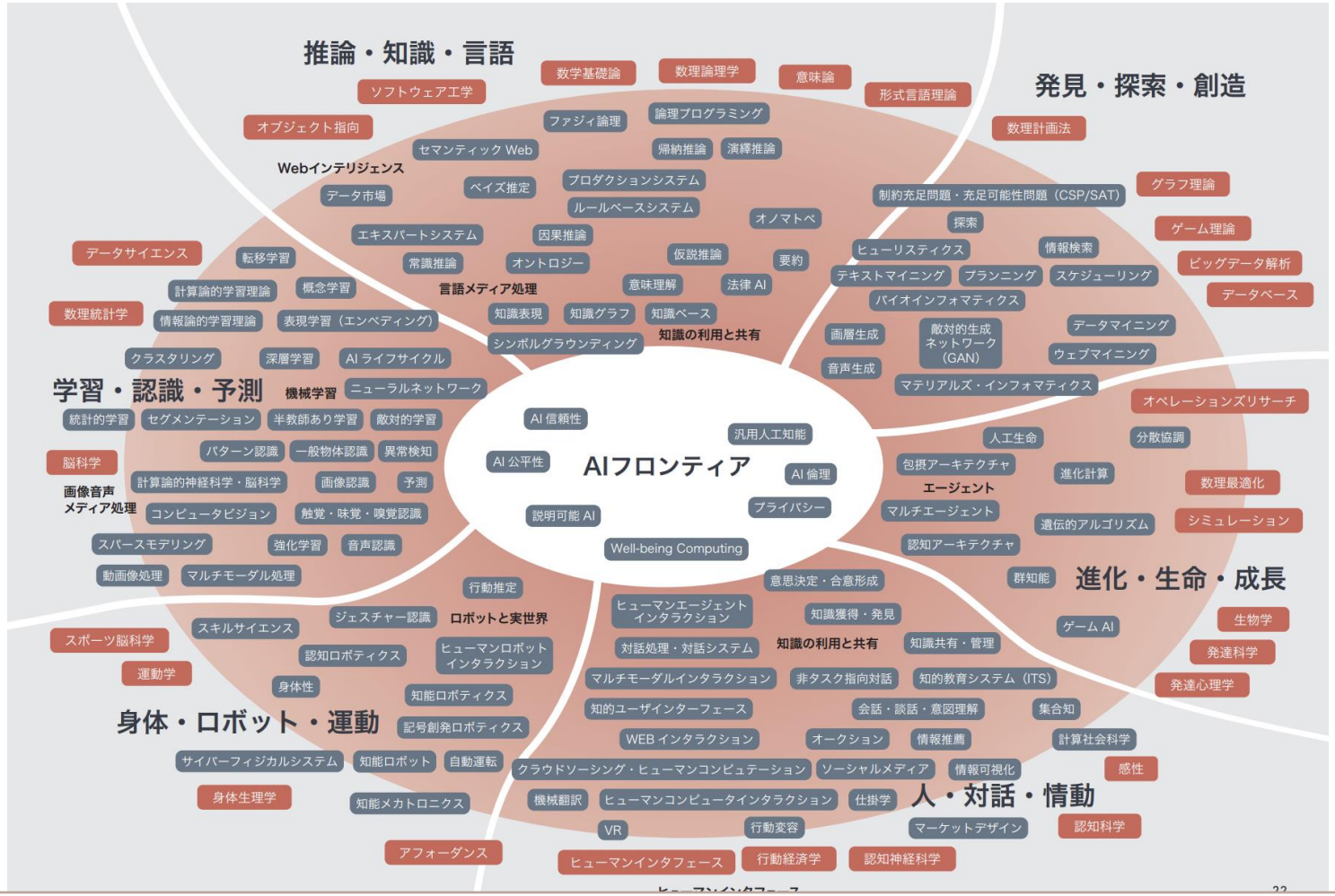
1. **数理最適化（解析的・厳密解）**
   - 数式を使って**理論的に最適解を導く方法**
   - 特徴：
     - 条件が単純なときに有効（線形計画法など）
     - 微分・方程式で解ける場合が多い
     - 例：線形計画法，ラグランジュ乗数法

2. **メタヒューリスティック（探索的・近似解）**
   - **自然界や経験則をヒントにした探索方法**
   - 特徴：
     - 複雑な問題にも使える
     - 厳密解でなくても「良い解」を見つけられる
     - 例：遺伝的アルゴリズム（GA），粒子群最適化（PSO），差分進化（DE）

AIの技術

人工知能学会 AIマップより

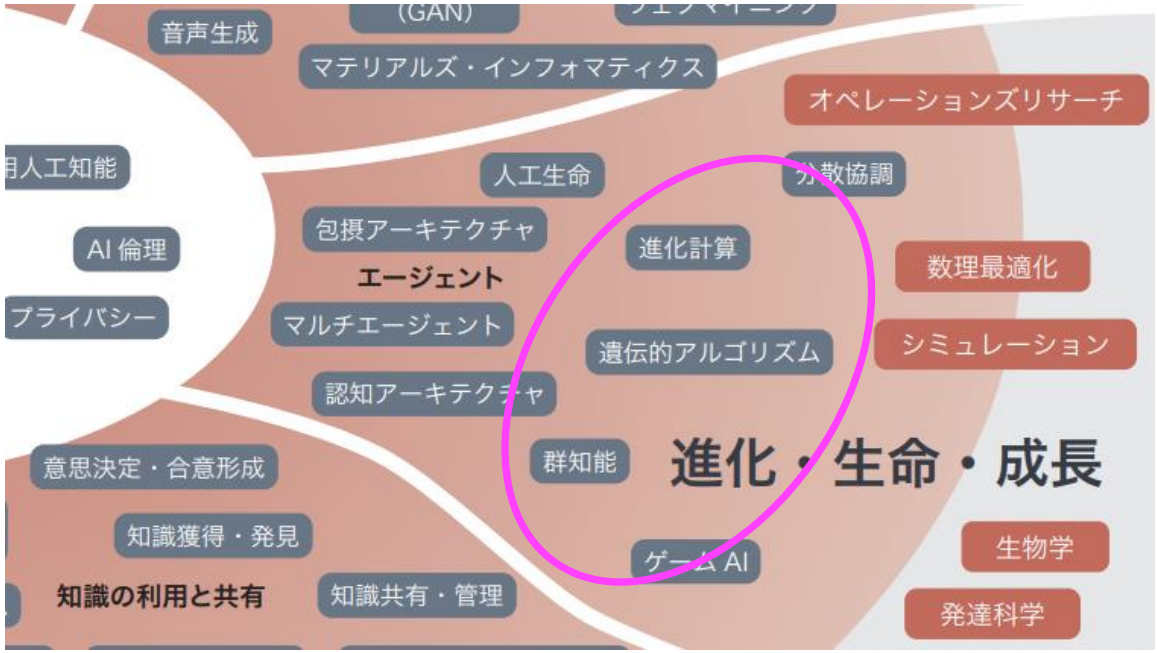
一般社団法人 人工知能学会
The Japanese Society for Artificial Intelligence



人工知能 とは「様々な技術の複合体の総称」

https://www.ai-gakkai.or.jp/resource/aimap/

メタヒューリスティック手法



- **進化計算**：生物の進化（自然選択や突然変異）を模倣した最適化手法の総称
- **遺伝的アルゴリズム（GA）**：進化計算の代表例で，遺伝子の交叉や突然変異を使って解を改良する手法
- **群知能**：動物の群れの協調行動をモデルにした最適化手法（例：PSO, ACO）

粒子群最適化
Particle Swarm Optimization (PSO)

PSO（粒子群最適化）

- 1995 年に提案された群知能ベースの最適化アルゴリズム
- **鳥の群れや魚の群れの社会的**行動をモデル化
- **複数の「粒**子」が情報を共有しながら協調して最適解を探索

PSOが対象とする問題の領域

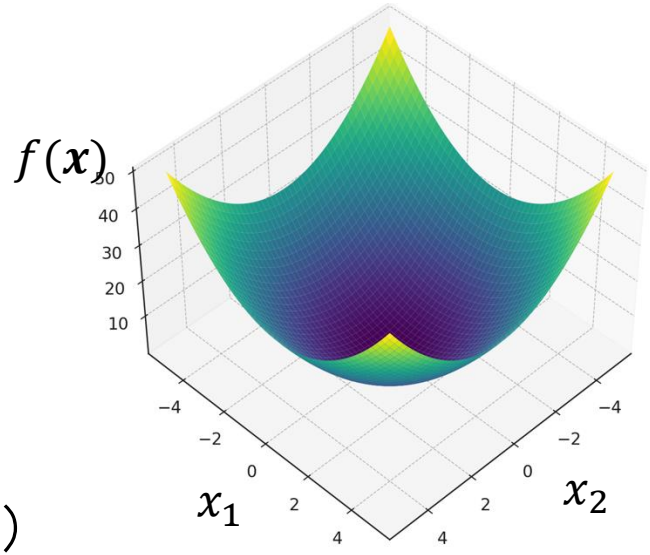
連続関数最適化問題（Continuous Optimization）

複数の連続変数を持つ関数の中で，
最も小さい値を与える入力を見つける

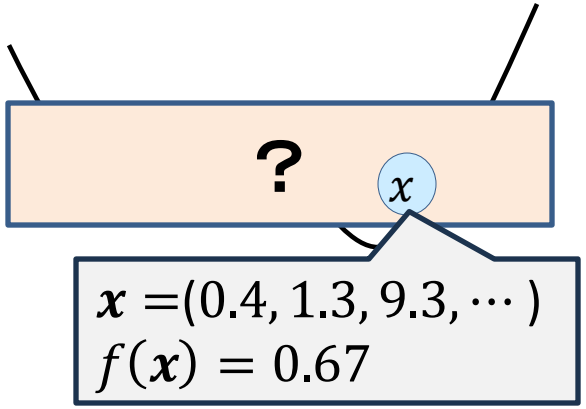
$$\min_{\boldsymbol{x}\in\Omega\subseteq R^n} f(\boldsymbol{x})$$

n は次元数，Rは実数

ただし，変数 x は実数の制約内で動くものとする



- 決定変数：$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$（実数ベクトル）
- 目的関数：$f(\boldsymbol{x})$（連続的に評価可能）
- 制約領域：Ω（例：範囲制約 $(l_i \leq x_i \leq u_i)$

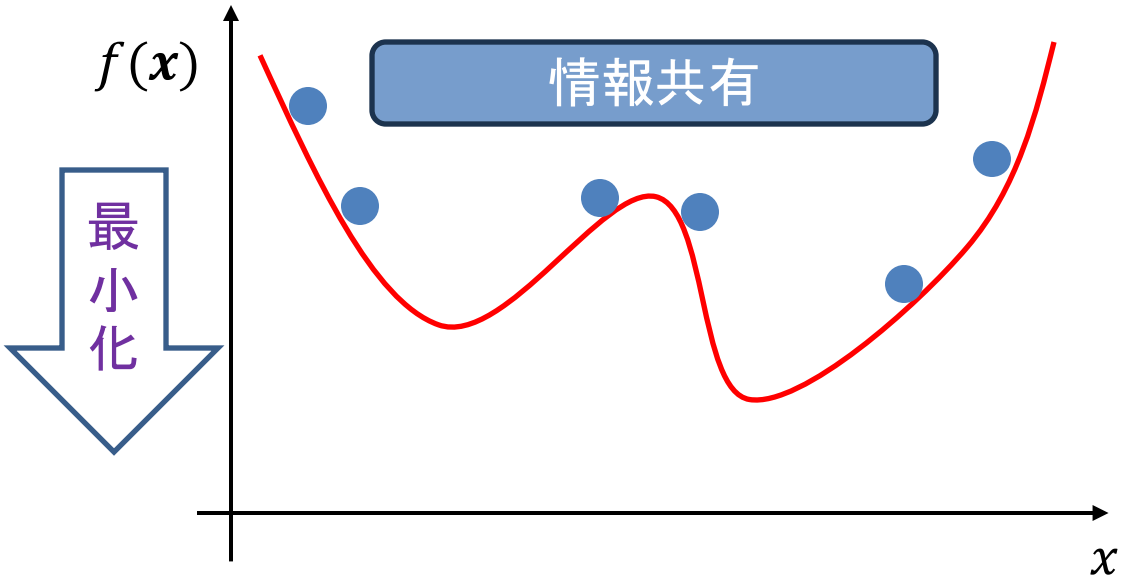


探索する関数の性質が分からなくても
（目的関数が不連続関数，微分不可能関数であっても）

探索点 $\boldsymbol{x}$ に対し，関数値 $f(\boldsymbol{x})$ を得ることができれば適用可能

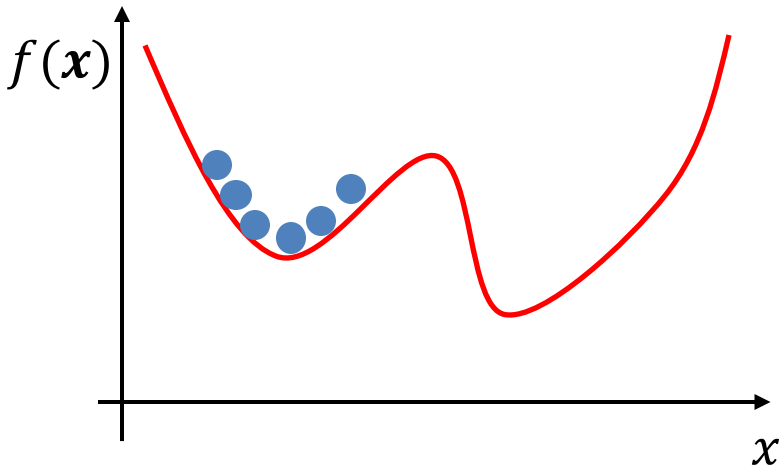
PSOの特徴 (1/2)

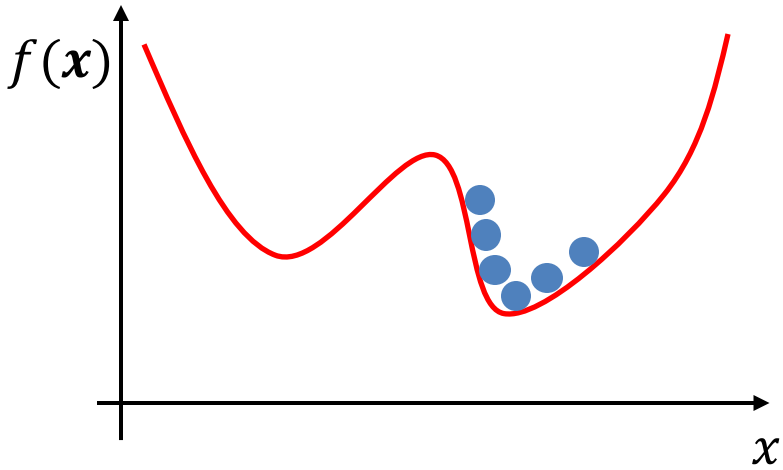
- 解集団を用いた多点探索，解候補を実数値ベクトルで表す
- 探索過程は確率的（乱数パラメータを含む）
  => 実行するたびに違う結果
- 集団内では粒子（個体）間にインタラクションがある
- 複数の粒子が存在して一定の性能を示す



PSOの特徴 (2/2)

- いくつかの制御パラメータが存在
  - 粒子数 N
  - 慣性係数 w
  - 認知パラメータ c_1
  - 社会パラメータ c_2
- 制御パラメータの値により，探索性能は大きく左右される
- 対象とする問題の性質により，適するパラメータは異なる

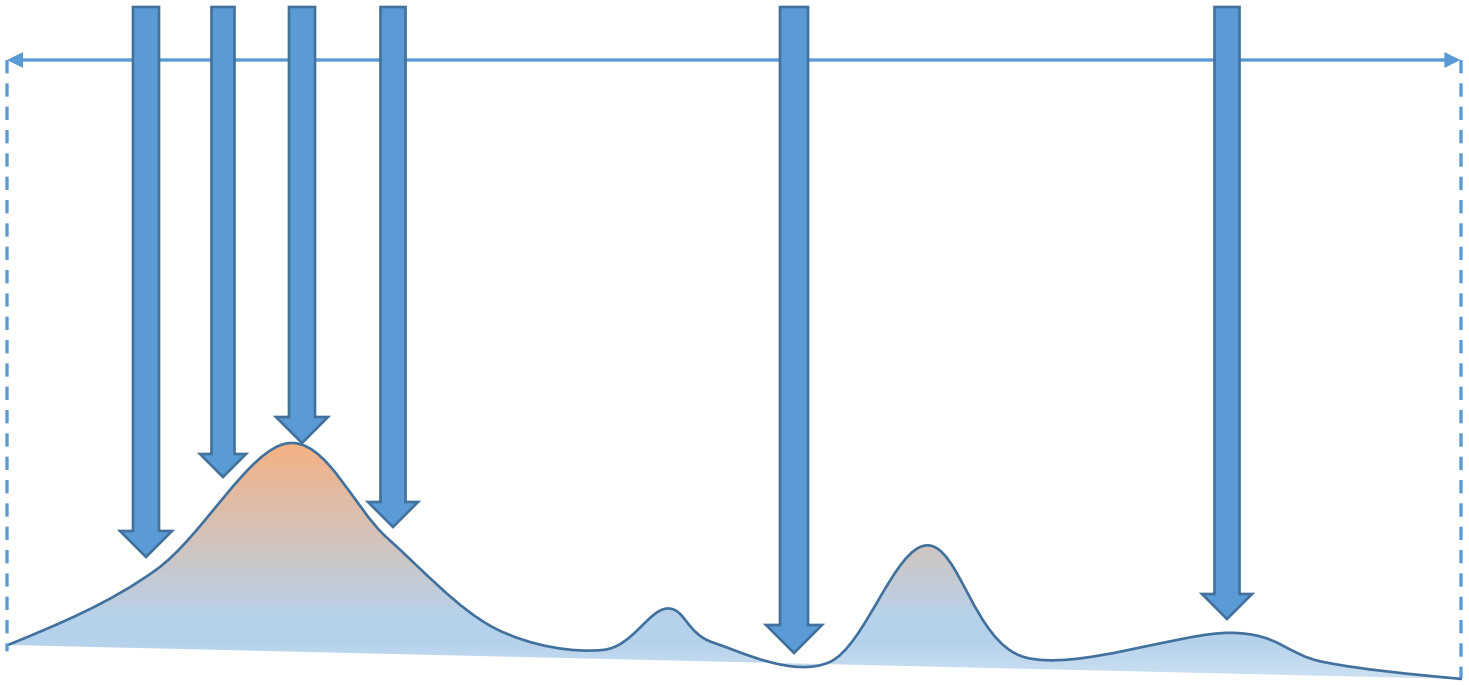




解探索のイメージ

関数の最小値(最大値)を探す＝砂漠で最も石油が出る場所を探す

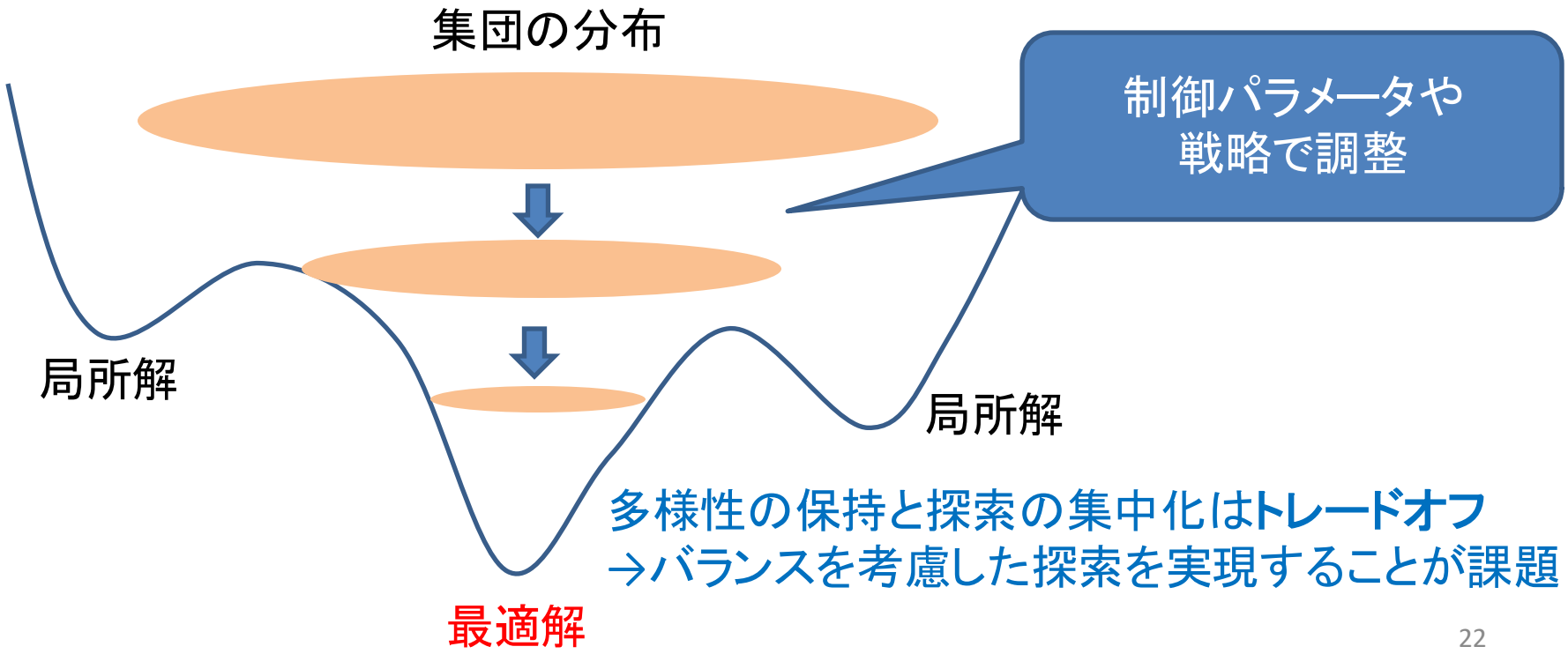
- 一回掘るとその場所の埋蔵量（良さ）が分かる
  - 掘ってみないとわからない！関数の形状は不明
- 何度も掘れば大体の埋蔵量が把握できるが，その分コストが掛かる
- なるべく少ない回数 (コスト) で，最も埋蔵量が多い場所を見つけるには？



理想的な解探索

大域的な探索から，局所的な探索へシフトする探索

- 探索序盤は十分な多様性を保持し，網羅的に広い範囲を探索
- 探索終盤は集団を収束させ，優良な解周辺を集中的に探索



PSOの処理手順

1. **初期化**
   - 粒子群 $P = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$ を定義，N は粒子数
   - 各粒子 i の位置 $\boldsymbol{x}_i$ はランダムに生成
2. **評価**
   - 各粒子 i の位置 $\boldsymbol{x}_i$ を目的関数（適応度関数）に入力し，良さ（適応度）$f(\boldsymbol{x}_i)$ を計算
3. **最良位置の記録**
   - $\boldsymbol{pbest}$：粒子ごとに「これまでで一番良かった位置」
   - $\boldsymbol{gbest}$：集団全体で「一番良かった位置」
4. **更新処理**
   - **速度更新**：粒子の進む方向を計算（$\boldsymbol{pbest}$と$\boldsymbol{gbest}$を参考にする）
   - **位置更新**：新しい位置に粒子を移動
5. **終了判定**
   - 最大ステップ数に達したら終了
   - そうでなければステップを進めて再び評価へ（2へもどる）
6. **結果出力**
   - 最終的な $\boldsymbol{gbest}$ を解として出力

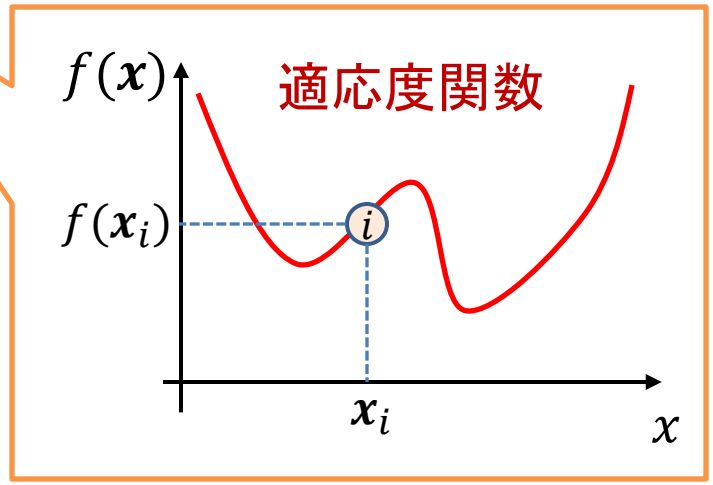
初期化時の処理：粒子群の生成

N は粒子数
D は問題の次元数

各粒子 i の持つ情報 $(i = 1, \cdots, N)$:

$\boldsymbol{x}_i$	x_1	x_2	$\cdots$	x_D	$f(\boldsymbol{x}_i)$
$\boldsymbol{pbest}_i$	p_1	p_2	$\cdots$	p_D	$f(\boldsymbol{pbest}_i)$
$\boldsymbol{v}_i$	v_1	v_2	$\cdots$	v_D	

D 次元ベクトル



各粒子 i について，
- 位置 $\boldsymbol{x}_i$ と速度 $\boldsymbol{v}_i$ を，あらかじめ決めた範囲内でランダムに決定
- 適応度 $f(\boldsymbol{x}_i)$ を計算
- 自身の最良位置 $\boldsymbol{pbest}_i$ は $\boldsymbol{x}_i$ のコピーとする
- 自身の最良位置の適応度 $f(\boldsymbol{pbest}_i)$ も $f(\boldsymbol{x}_i)$ のコピーとする

$f(\boldsymbol{pbest}_i)$ のうち最良のものを，群れ全体の最良位置 $\boldsymbol{gbest}$ とする

PSOの疑似コード

Algorithm of PSO

・初期個体の生成

for $t = 1$ to T_{max} do

  for $i = 1$ to N do

    ・速度と位置の更新

    ・粒子の評価

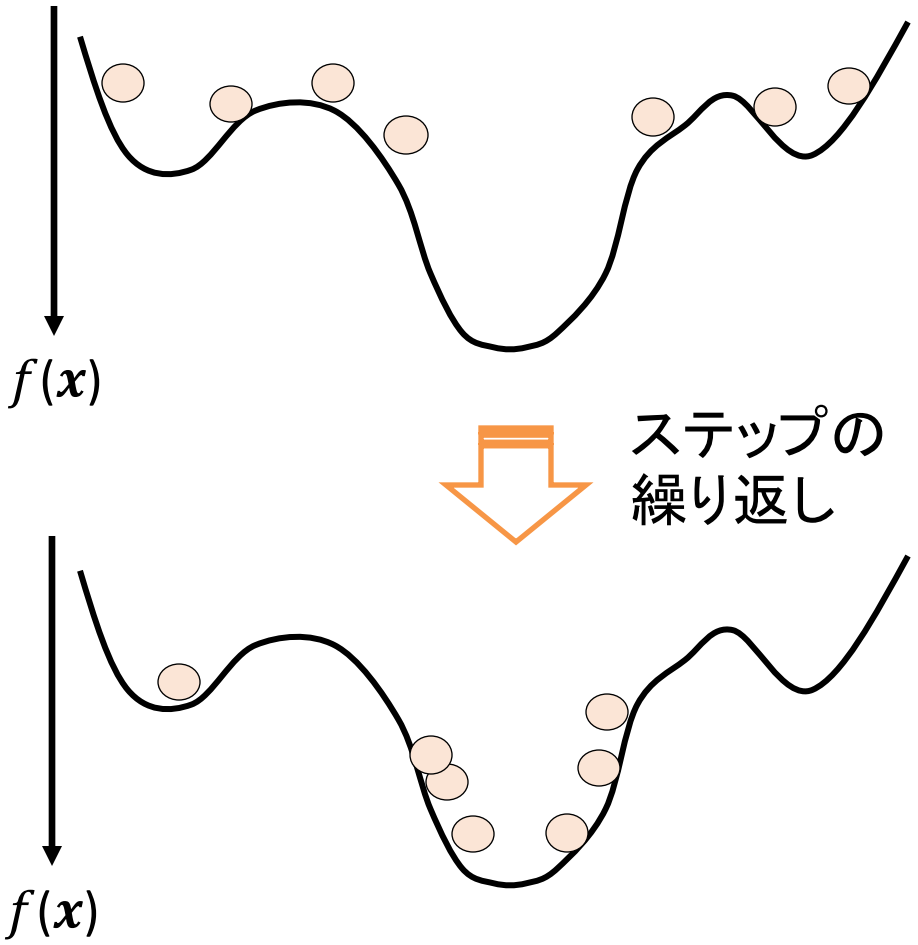
    ・p(g)best の更新

  end for

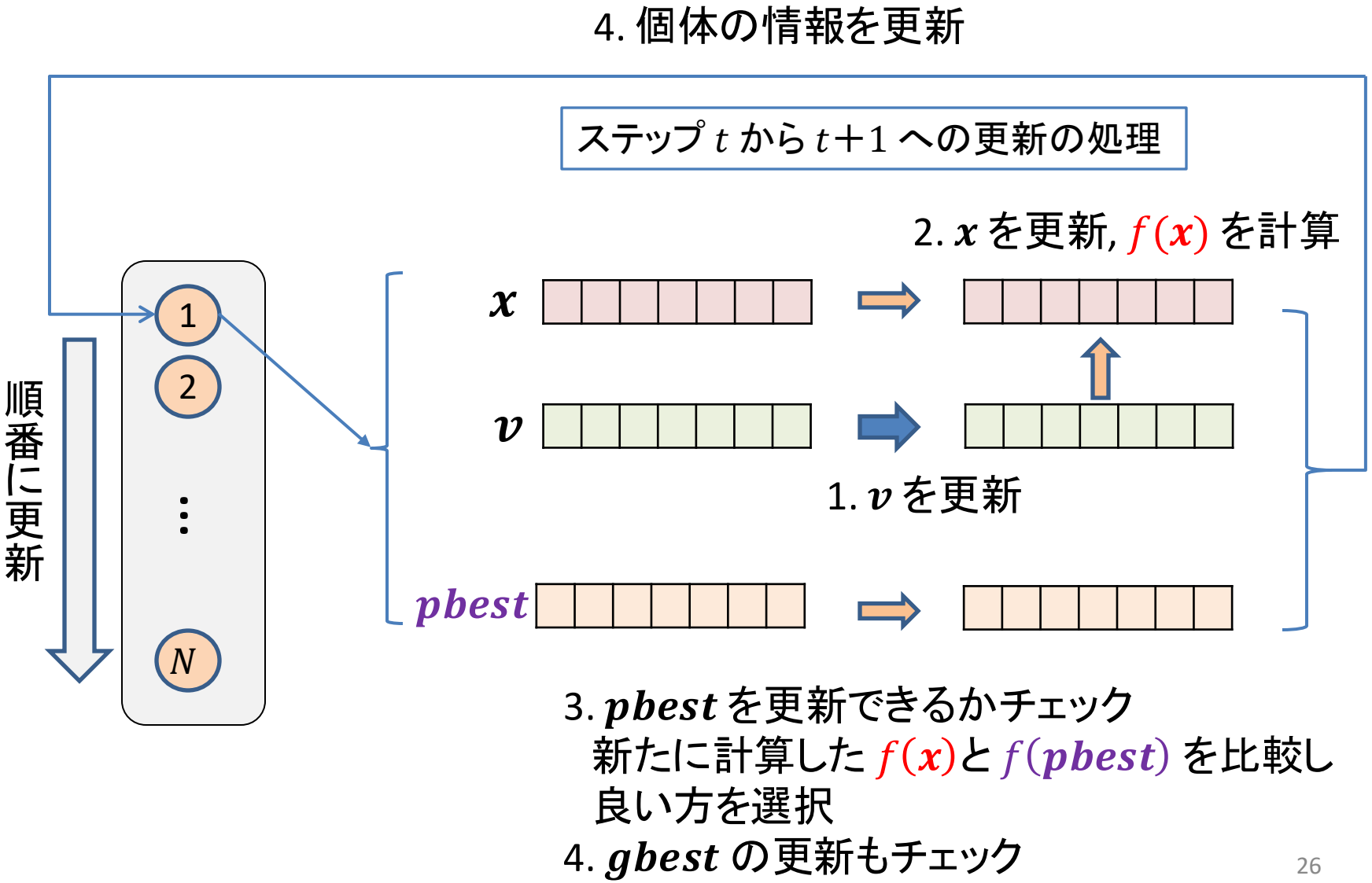
end for

粒子のループ

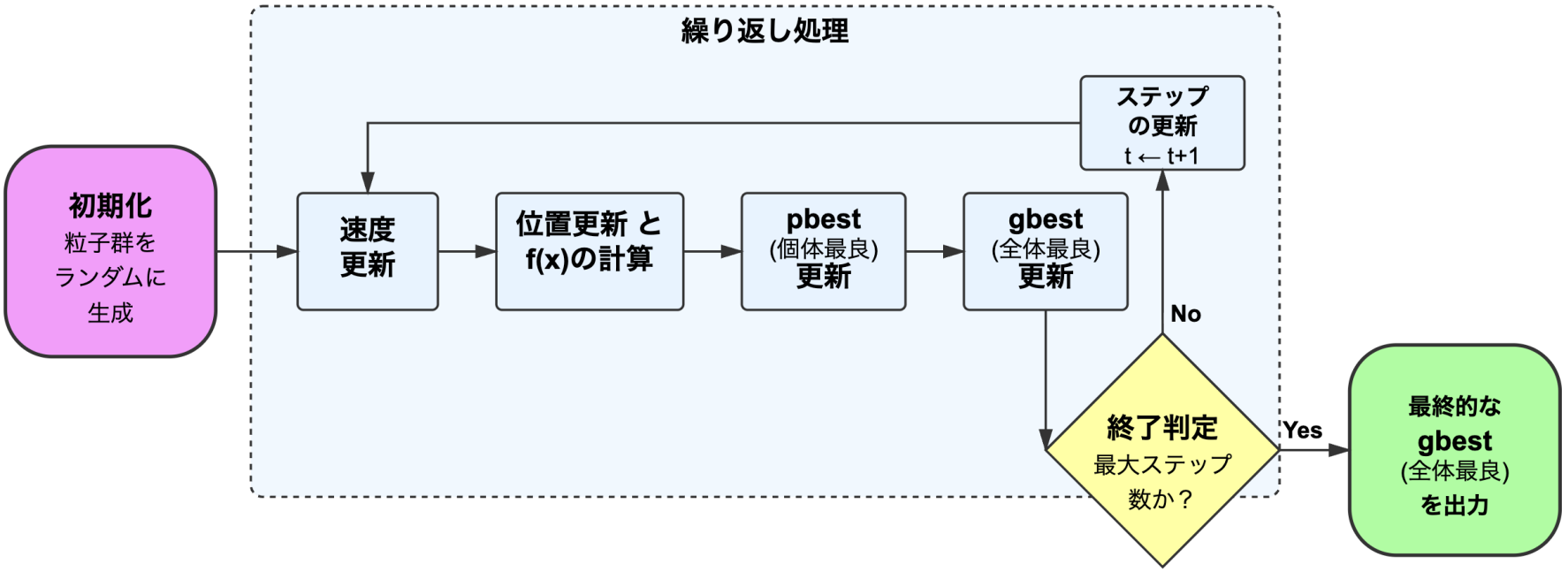
ステップのループ

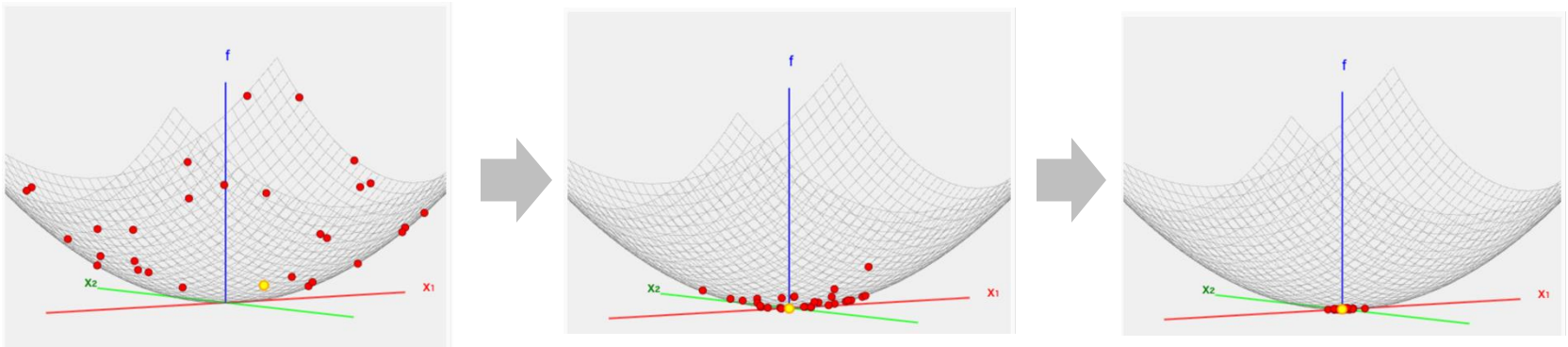


各ステップの処理


4. 個体の情報を更新
ステップ t から $t+1$ への更新の処理
2. x を更新, $f(x)$ を計算
x
v
1. v を更新
$pbest$
順番に更新
1
2
N
3. $pbest$ を更新できるかチェック
新たに計算した $f(x)$ と $f(pbest)$ を比較し
良い方を選択
4. $gbest$ の更新もチェック

PSOの処理手順と探索のイメージ





初期化時　　探索中盤　　探索終盤

粒子の位置の更新 1/2

t ステップでの，粒子 i の j 次元の成分の更新

$$v_{ij}^{t+1} = \omega v_{ij}^{t} + c_1 rand_{1ij}\left(x_{ij}^{*} - x_{ij}^{t}\right) + c_2 rand_{2ij}\left(x_{Gj}^{*} - x_{ij}^{t}\right)$$

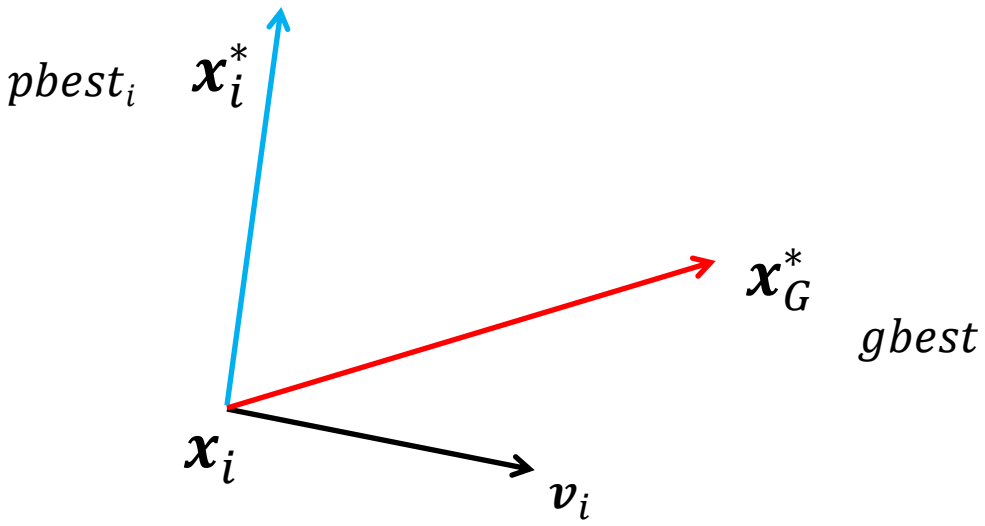
$$x_{ij}^{t+1} = x_{ij}^{t} + v_{ij}^{t+1}$$

自己の最良位置に向かうベクトル

群れの最良位置に向かうベクトル

- $rand_{1ij}$ と $rand_{2ij}$: 各粒子の次元毎に生成される [0,1]の一様乱数
- ω, c_1, c_2 は慣性係数，認知パラメータ，社会パラメータ

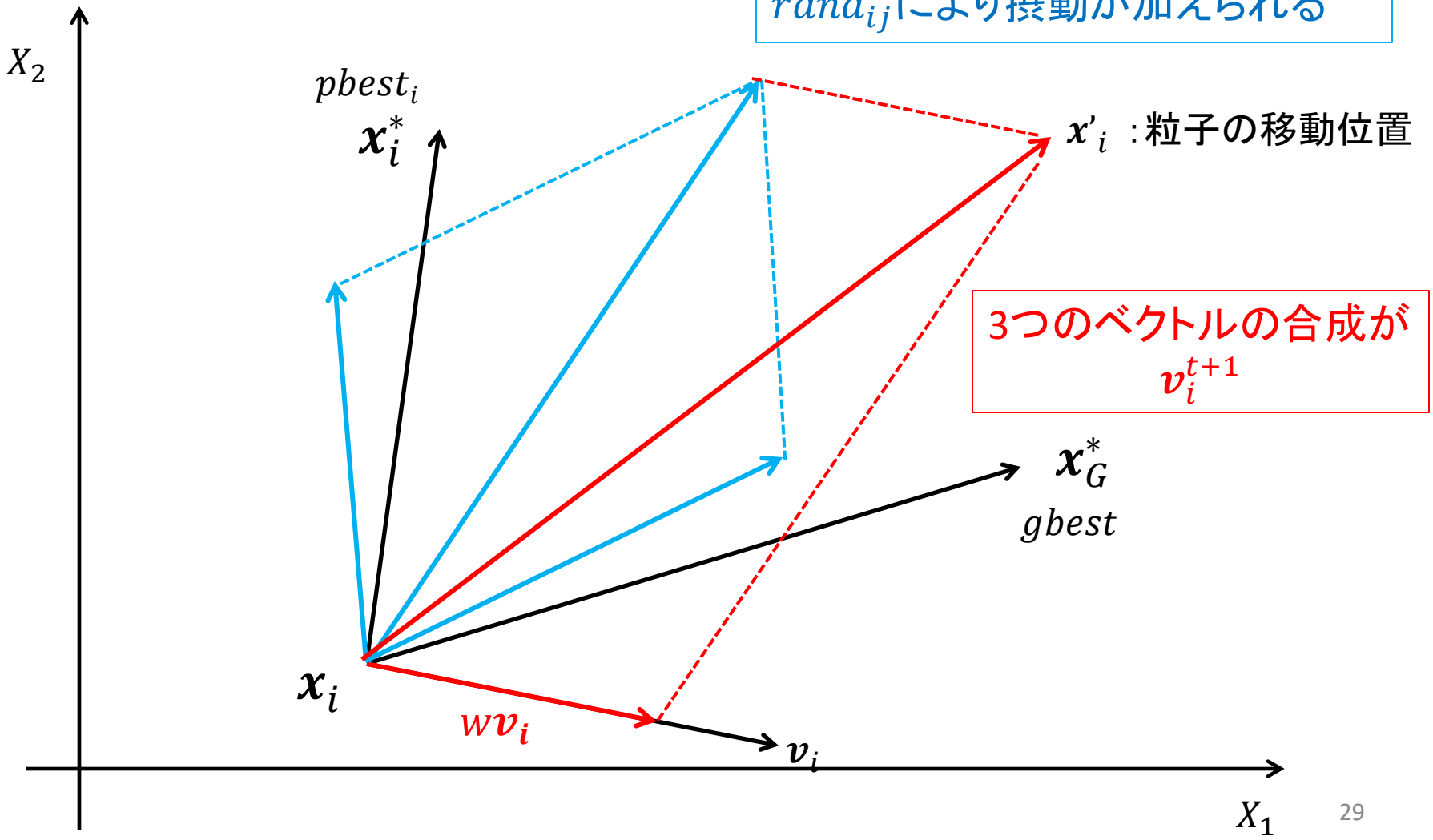
ω (小文字のオメガ)は
プログラムだとW



粒子の位置の更新 2/2

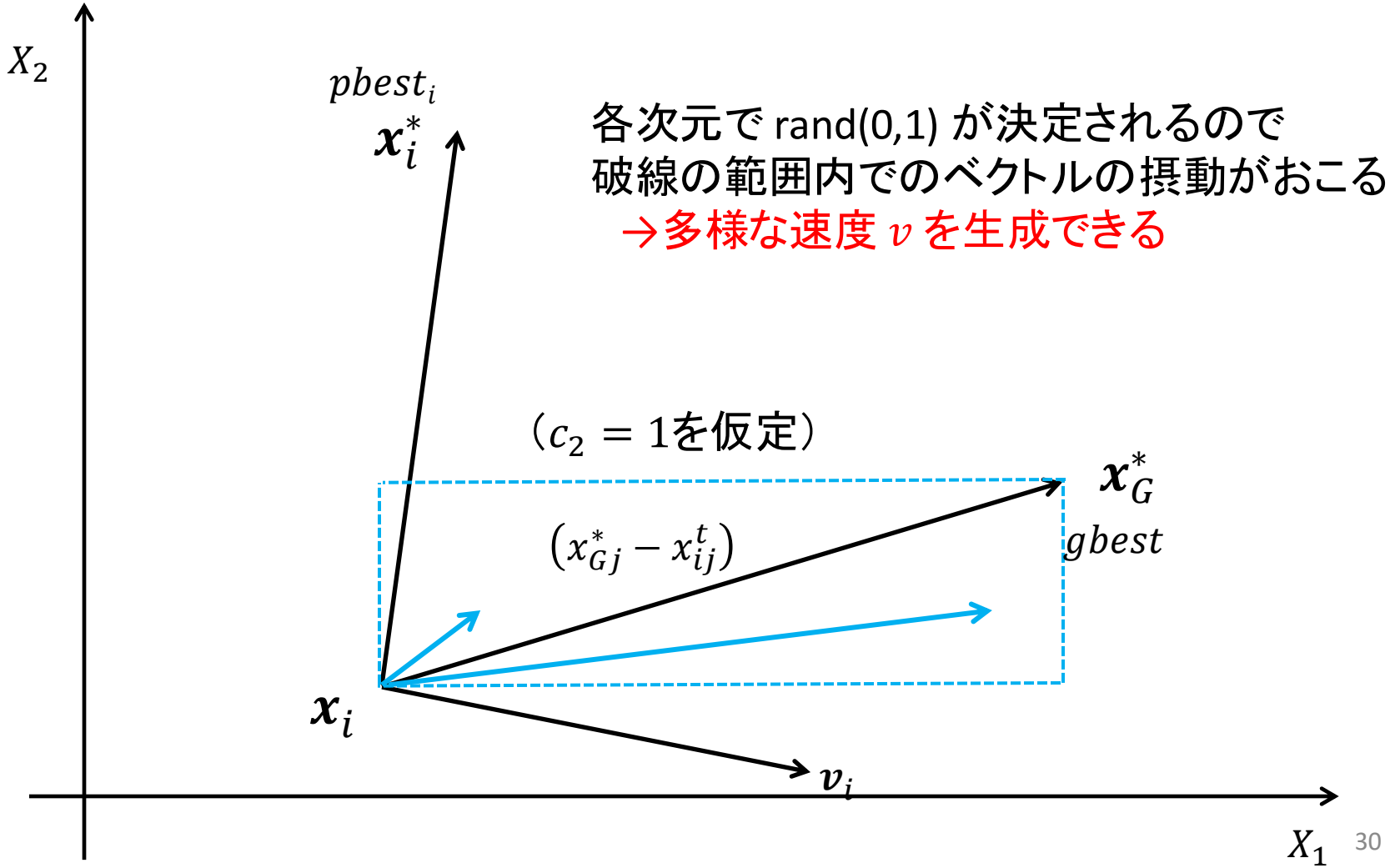
$$v_{ij}^{t+1} = \omega v_{ij}^{t} + c_1 rand_{1ij}\left(x_{ij}^{*} - x_{ij}^{t}\right) + c_2 rand_{2ij}\left(x_{Gj}^{*} - x_{ij}^{t}\right)$$

pbest と gbest へ向かうベクトルは
$c_{\{1,2\}}$倍されて
$rand_{ij}$により摂動が加えられる



$rand$の解探索への影響

$$v_{ij}^{t+1} = \omega v_{ij}^{t} + c_1 rand_{1ij}\left(x_{ij}^{*} - x_{ij}^{t}\right) + c_2 rand_{2ij}\left(x_{Gj}^{*} - x_{ij}^{t}\right)$$



粒子群の安定･不安定

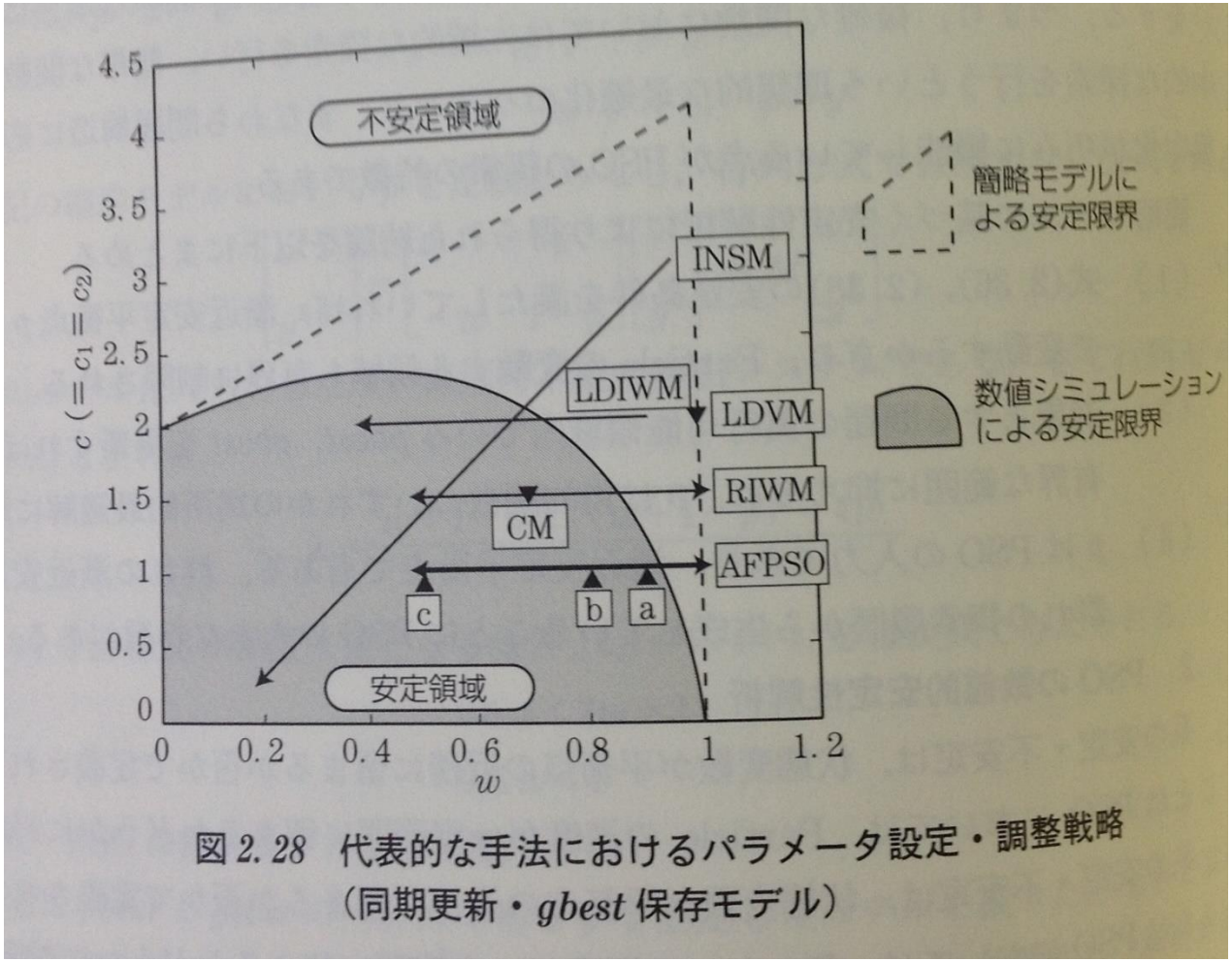


図 2.28　代表的な手法におけるパラメータ設定・調整戦略
（同期更新・*gbest* 保存モデル）

メタヒューリスティクスと応用，相吉&安田，電気学会, 2007, （82Pより）