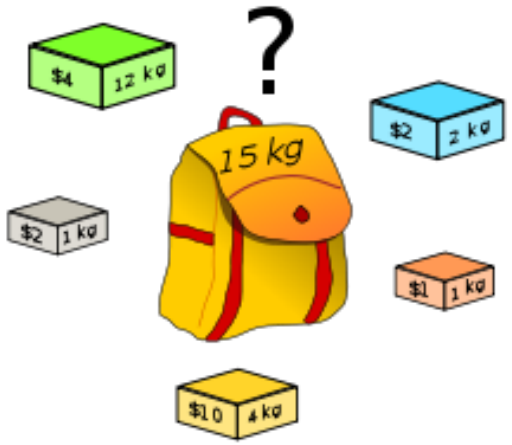
計算知能
(COMPUTATIONAL INTELLIGENCE)

第２回　最適化の事例

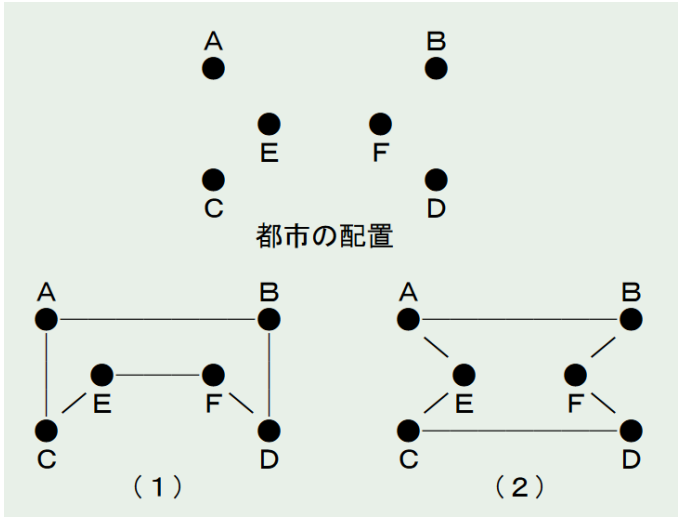
教員：　谷口彰

今回の内容

- 組合せ最適化問題
  - 組み合わせ最適化問題の典型例
  - 欲張り法
  - 粒子群最適化法



ナップサック問題



巡回セールスマン問題

復習：
最適化問題

$$\min_{x} f(x) \quad \text{←目的関数}$$
$$\text{s.t.}\ x \in F \quad \text{←制約条件}$$
$$F \subseteq X$$

- x：決定変数（対象問題で決定すべき量）
  一般には複数あるので、ベクトルで表現
  （そのベクトル空間を X とする）
- $f(x)$：目的関数（ x の良さ/悪さを与える値）
- F：可能解領域（空間 X の中で解が許される範囲）

最適化 ： 目的関数の最大化/最小化

復習：
問題の分類、難しさのレベル(1/2)

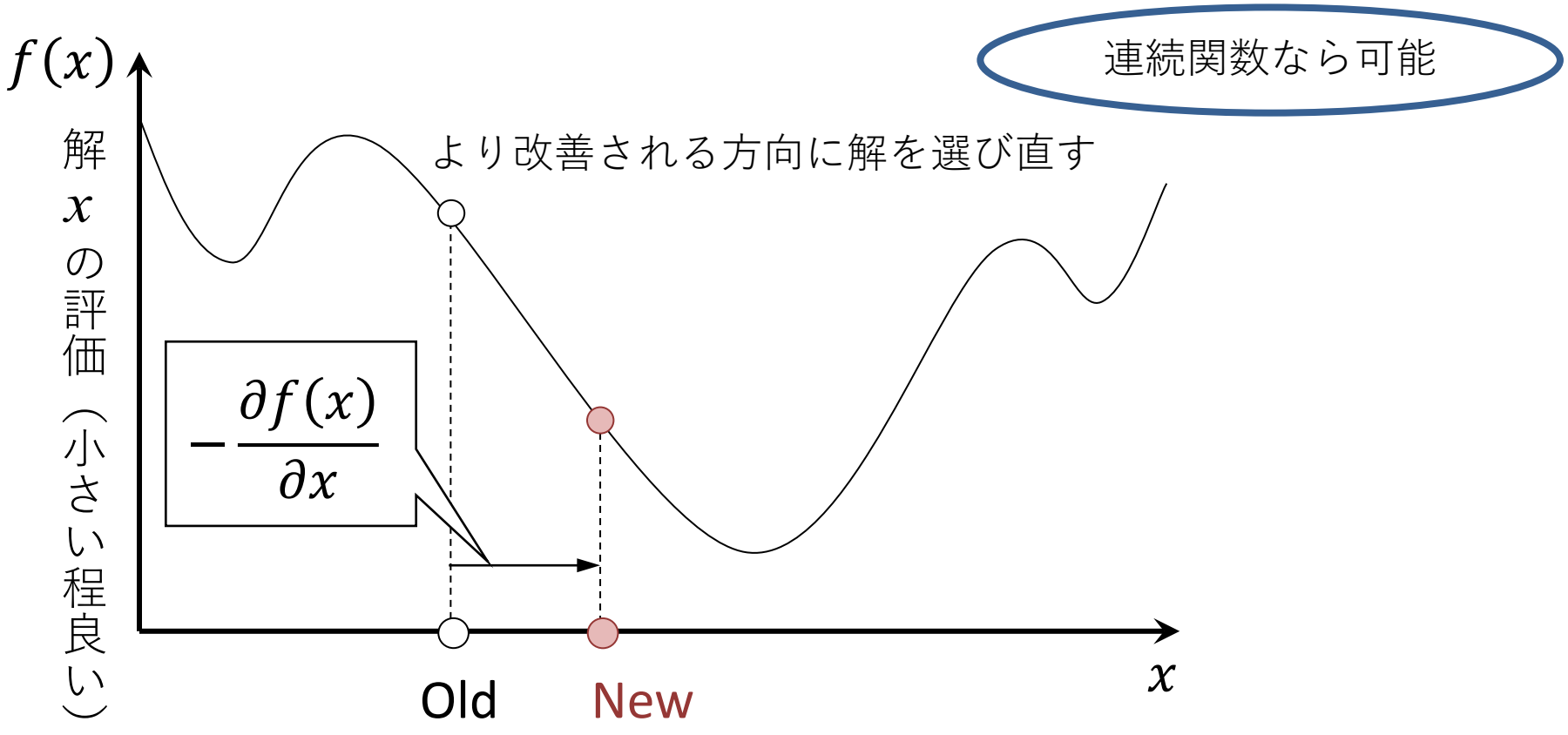
- 解ける（数学的or解析的に最適解を求められる）場合
  ベクトル空間 X がn次元の実数値ベクトルで、
  F が綺麗な形（n次元凸集合）　←凸多角形の線形計画問題

- 厳密には解けないが、比較的単純な探索手続きがある
  （F がn次元凸多角形でない　etc.）場合
  しかし、微分操作はできる
  →　ex.最急降下法で局所最適を何度も繰り返し求めて、
  　　その中に本当の最適解が入っていることを期待する）

- 単純な手続きがなく、全解を列挙するしかない場合

復習：最急降下法（微分を使う局所探索）

- 近くのもの（xの近傍）は、良さの度合いも近い（$f(x)$が連続）
- 連続ならば、微分が使える（今のxからどちらの方向に行けばより良くなるか、が分かる）



復習：焼きなまし法

- 温度 T は最初は高く、例えば探索回数に応じて減少させる
  （探索初期：高 → 探索終期：低）
- 解空間の形状があらかじめ判明している場合、
  探索初期：最大の山を超えられるように（局所解に陥らないように）、
  探索終期：大きな山は越えず、局所的な最適解を取得できるように

- 探索初期：　大域的に（大雑把に）良い解を探索
- 探索中期：　ある程度近傍に限定して、解を探索
- 探索終期：　（ほぼ探索が完了しているものとして）
  近傍の最適解を厳密に探索

それぞれに異なる目的を持ち、異なる探索を実施

非常に多くの手法に応用されている

復習：
問題の分類、難しさのレベル(2/2)

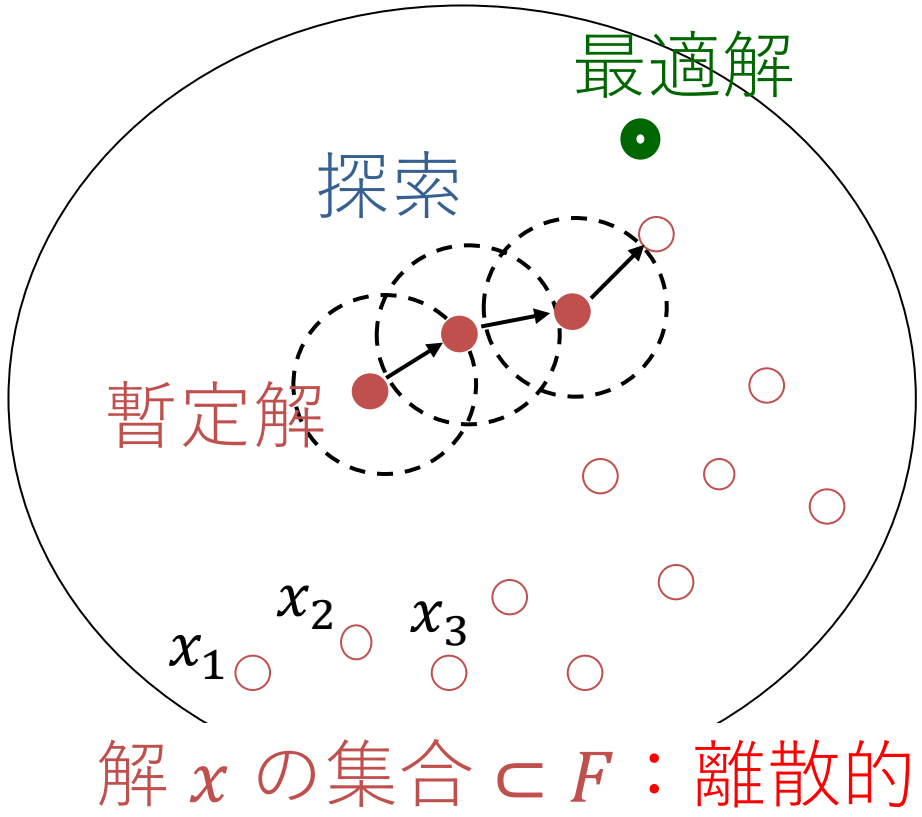
- 解ける（数学的or解析的に最適解を求められる）場合
  ベクトル空間 X がn次元の実数値ベクトルで、
  F が綺麗な形（n次元凸集合）　←凸多角形の線形計画問題

- 厳密には解けないが、比較的単純な探索手続きがある
  （F がn次元凸多角形でない　etc.）場合

- 単純な手続きがなく、全解を列挙するしかない場合
  - 変数xが離散的 → 頑張って探索
    ex.組合せ最適化問題
  - $f(x)$が不連続 → 不連続点では探索の手掛かりもない

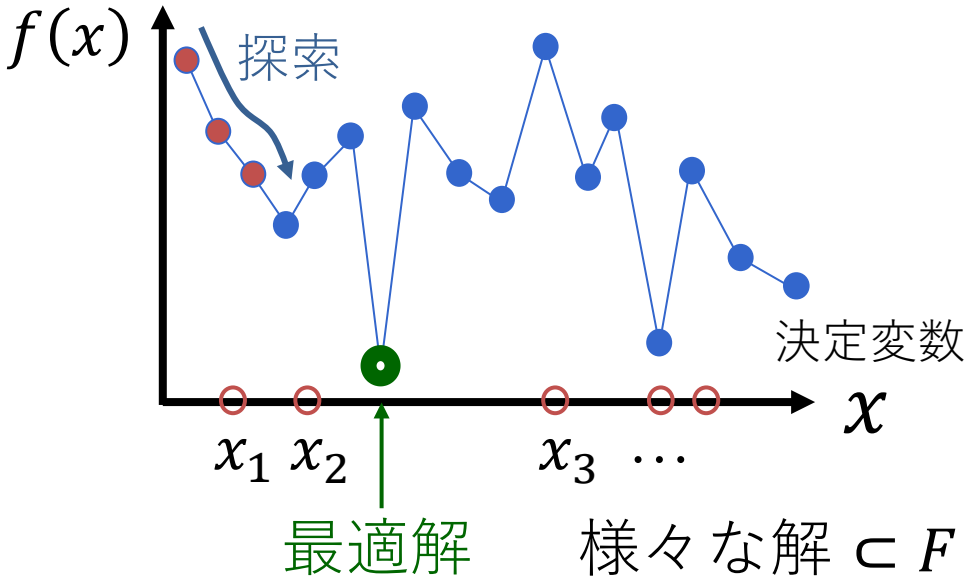
問題に応じて対応策を検討

最適解の探索



解 x の集合 $\subset F$：離散的

目的関数 ← 解の良さ（最小化なら低さ）



解がより改善される方向に、様々な
解の集合（解空間）の中を探しに行く

組合せ最適化問題の解き方

- 基本的に、列挙する（全ての解を挙げる）ことでしか最適解は得られない

- でも解の数は非常に多い（NP困難な問題）

- 厳密な最適解が必要か？
  - 厳密解を得たい → すべて列挙するしかない

厳密解法＝効率的な全列挙法：分枝限定法、動的計画法

  - 近似的な解（最適に近そうな解）で妥協する

近似解法：部分的列挙法／発見的方法／探索的方法

欲張り法

遺伝的アルゴリズム、
粒子群最適化法

欲張り法

- 貪欲法、グリーディ算法ともいう
- 欲張り法は近似アルゴリズムの一種
- 欲張り法は問題の要素を複数の部分問題に分割し、それぞれを独立に評価し、評価値のもっとも高いものを取り込んで解を得る手法

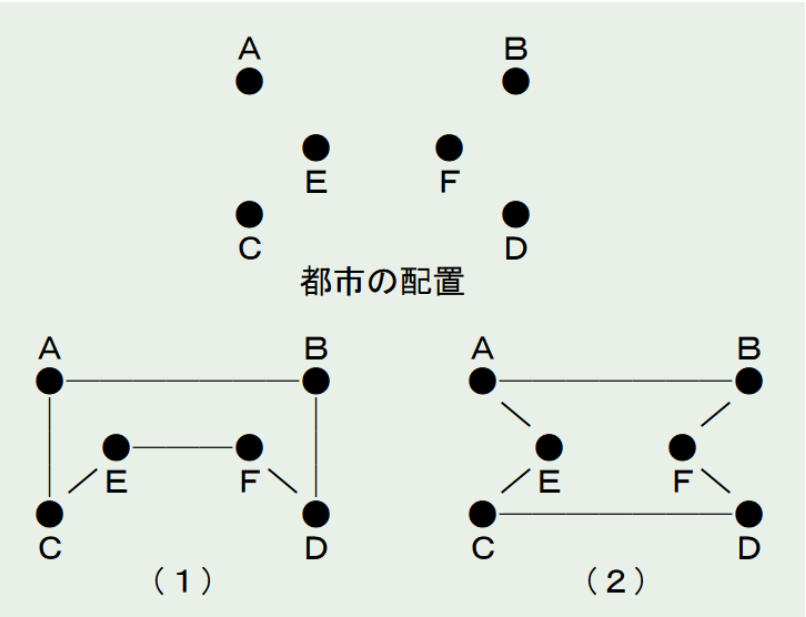
組合せ最適化問題の典型例
（色々な問題がこれらのいずれかになっていることが多い）

- 巡回セールスマン問題
- ナップザック問題
- １機械スケジューリング問題
- 最大充足可能性問題と充足可能性問題
- 一般化割当問題
- グラフ彩色問題
- 整数計画問題

巡回セールスマン問題（TSP）(1/3)

■ 巡回セールスマン問題：

N個の都市があり、あるセールスマンが各都市を一度ずつ訪問しなければならない。各都市の位置（あるいは各都市間の距離）はわかっている。このとき、巡回経路の総距離が最小になるような巡回路を求めよ。

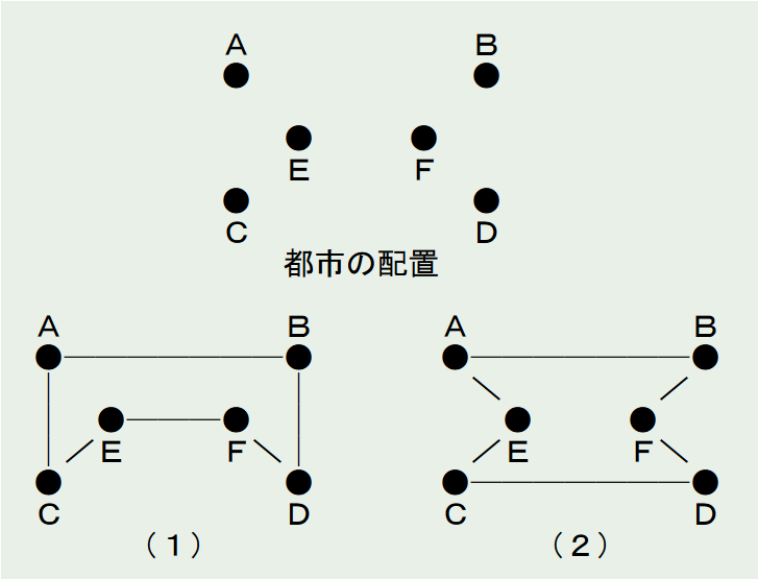


巡回セールスマン問題のイメージ

巡回セールスマン問題（TSP）(2/3)

- 出発点を A とすると(1) の巡路は A - B - D - F - E - C - A となり、
  (2) の場合は A - B - F - D - C - E - A となる

- 都市の個数は 6 つあるので巡路の総数は (6 - 1)! ＝ 120 通り
  （円順列と同じ） となる
  （逆回りの経路を省くと 120 / 2 ＝ 60 通り）



巡回セールスマン問題（TSP）(3/3)

- しかし、n 個の都市の場合は (n - 1)! / 2 個の巡路を調べる必要がある

巡路の総数が n の階乗に比例して増えるというのは爆発的

n	n^2	2^n	n !
1	1	2	1
2	4	4	2
3	9	8	6
4	16	16	24
5	25	32	120
6	36	64	720
7	49	128	5040
8	64	256	40320
9	81	512	362880
10	100	1024	3628800
.	.	.	. . .
15	225	32768	1307674368000
.	.	.	. . . . .
20	400	1048576	2432902008176640000

全部調べることができない

違う考え方が必要

TSP を「欲張り法」で解く(1/6)

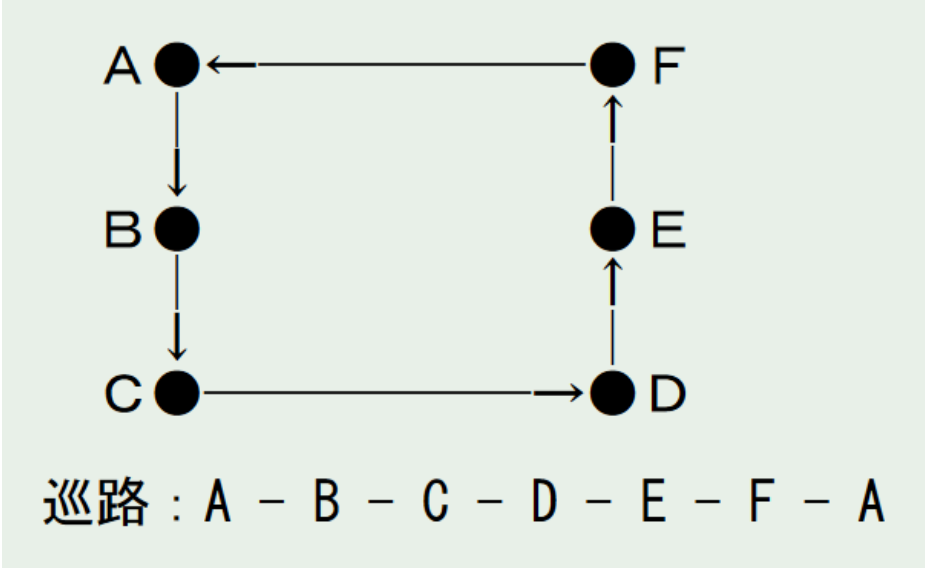
■ 巡回セールスマン問題：

N個の都市があり、あるセールスマンが各都市を一度ずつ訪問しなければならない。各都市の位置（あるいは各都市間の距離）はわかっている。このとき、巡回経路の総距離が最小になるような巡回路を求めよ。

欲張り法を使用して、巡回セールスマン問題を解く

TSP を「欲張り法」で解く(2/6)

■ 欲張り法では経路すべてを考えるのではなく、
まずは出発点から順番に最も近い都市を選んでいく



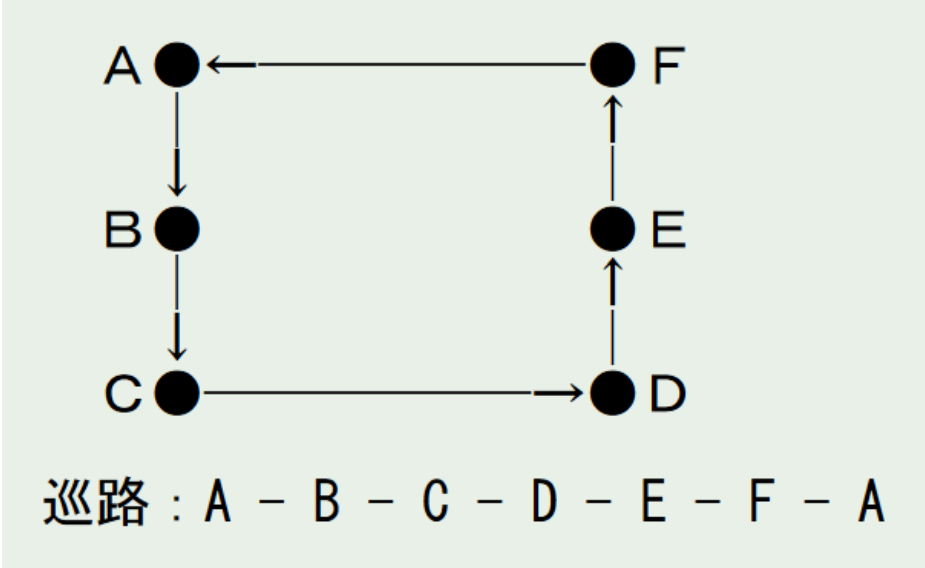
欲張り法による巡路の形成

TSP を「欲張り法」で解く(3/6)

- 出発点を A とすると、次は A にいちばん近い都市 B を訪問
- その次は B にいちばん近い都市 C を訪問
- このように、現在地点から最も近くにある都市を選択

つまり、後のことを考えずに「最短距離にある都市」から選んでいく

ところが「欲張り」と呼ばれる理由

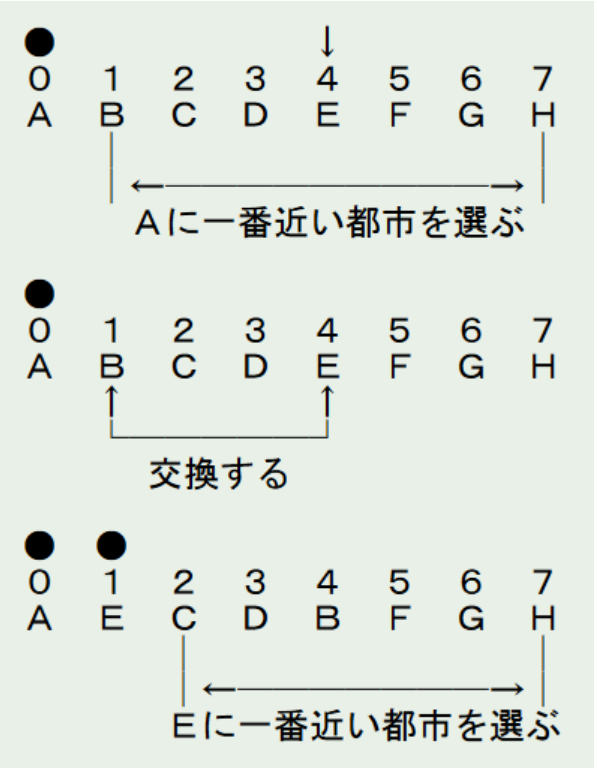


TSP を「欲張り法」で解く(4/6)

- このアルゴリズムのポイントは、未訪問の都市の中から最短距離にある都市を選択するところ
- 未訪問の都市を区別するため、配列内のデータを移動することで、訪問済みと未訪問の都市を区別する

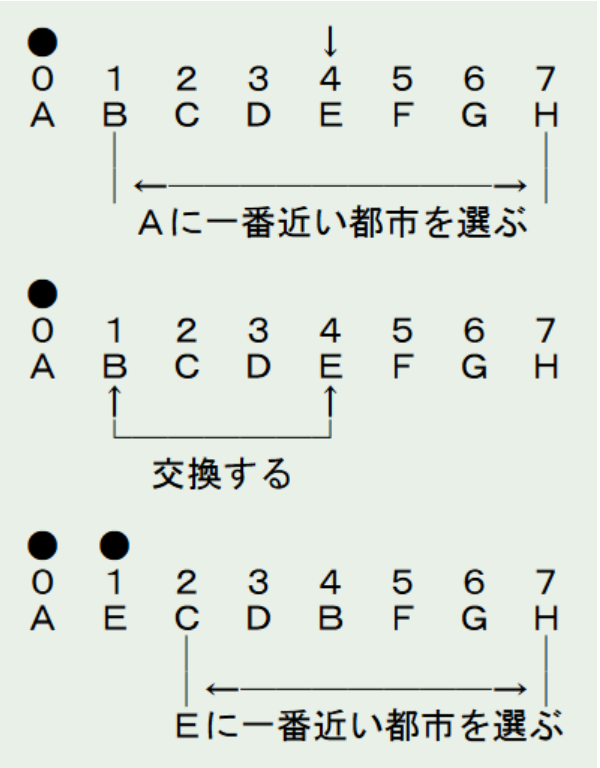
TSP を「欲張り法」で解く(5/6)

- 図に示すように、８つの都市が配列に格納されている
- 出発点を A とすると、次に訪問する都市は残りの都市（1 から 7 に格納されている都市）の中から A に一番近い都市



TSP を「欲張り法」で解く(6/6)

- これが E であったとし、この E を A の次の位置にある都市 B と交換する
  すると、0 と 1 にある都市 A と E は訪問済みで、そのほかの都市が
  未訪問と簡単に区別可能



TSP を「焼きなまし法」で解く(1/3)

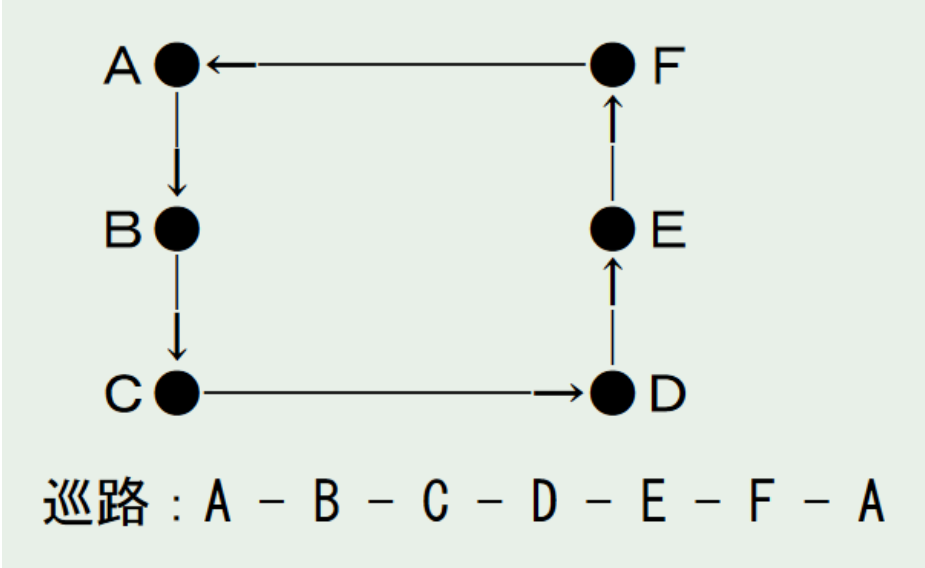
■ 巡回セールスマン問題：

N個の都市があり、あるセールスマンが各都市を一度ずつ訪問しなければならない。各都市の位置（あるいは各都市間の距離）はわかっている。このとき、巡回経路の総距離が最小になるような巡回路を求めよ。

焼きなまし法を使用して、巡回セールスマン問題を解く

TSP を「焼きなまし法」で解く(2/3)

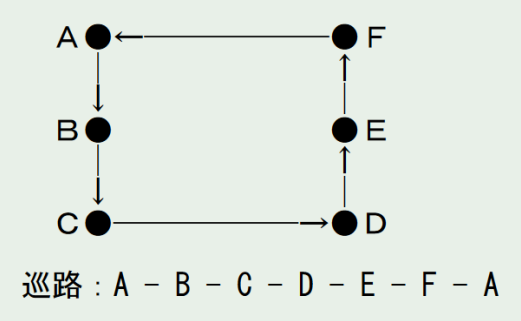
- まずはランダムに順路を決定したうえで、これを初期値として焼きなまし法を使用して探索を行う。
  同時に温度 T も設定する



欲張り法による巡路の形成

TSP を「焼きなまし法」で解く(3/3)

- 探索初期：大域的に（大雑把に）良い解を探索
  → 多数の順路を変更しながら、探索を実行
- 探索中期：ある程度近傍に限定して、解を探索
  → 中程度の順路を変更しながら、探索を実行
- 探索終期：（ほぼ探索が完了しているものとして）
  近傍の最適解を厳密に探索
  → 少数の順路を変更しながら、探索を実行



例えば、5つの順路の内
- 探索初期：隣り合う3か所を変更
- 探索中期：隣り合う2か所を変更
- 探索終期：隣り合う1箇所のみ変更

同時に、探索回数に応じて温度を下げる

5 分休憩

粒子群最適化法(1/8)

- 昆虫の大群や魚群において、一匹が良さそうな経路を発見すると（すなわち、食料を発見したとか安全であるという場合）、群れの残りはどこにいても素早くそれにならうことができる

⇩

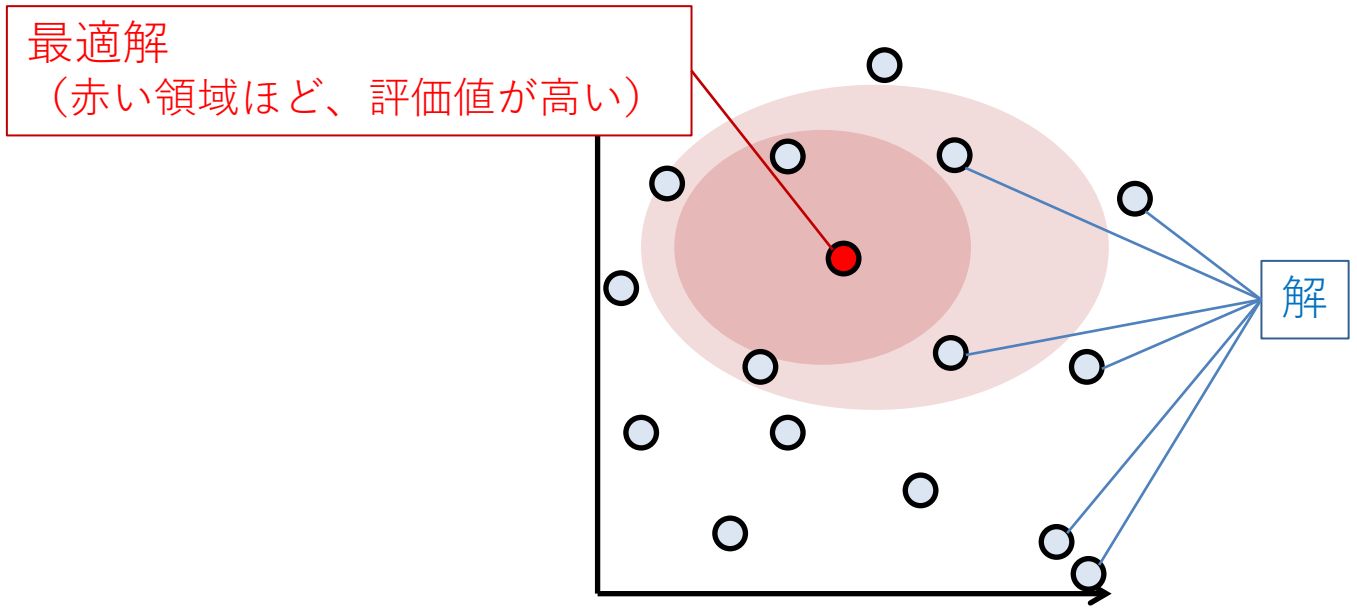
多次元空間において、位置と速度を持つ粒子群でモデル化

複数の粒子（点）で探索を行う、多点探索手法

粒子群最適化法(2/8)

- 群（解集団）に含まれる複数個の探索個体が、群で情報を共有しながら最良値を探索

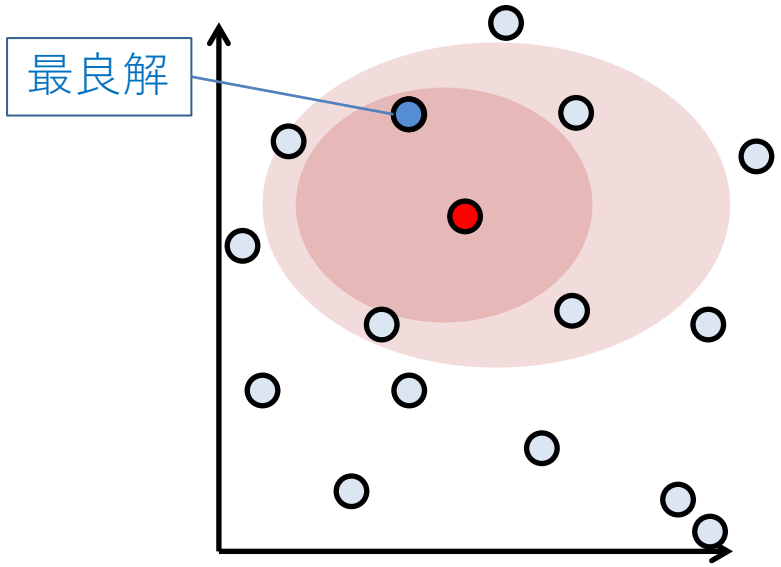
1. 解空間に群（解集団）をランダムに配置



粒子群最適化法(3/8)

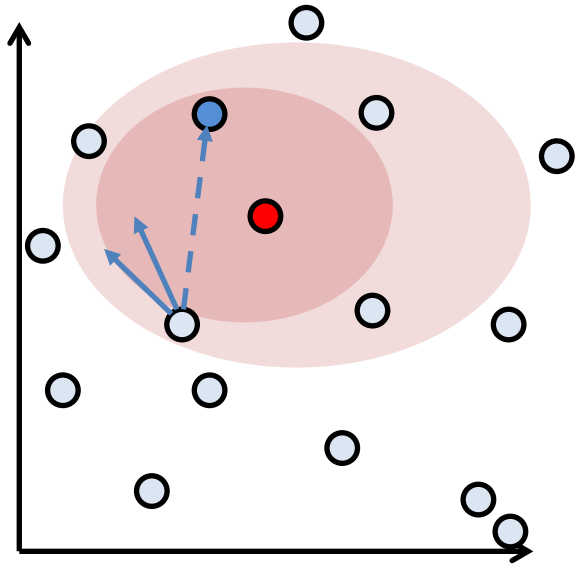
2. 解集団の適応度を評価し、最良解を決定

（最良解の適応度が十分であれば、探索を終了）



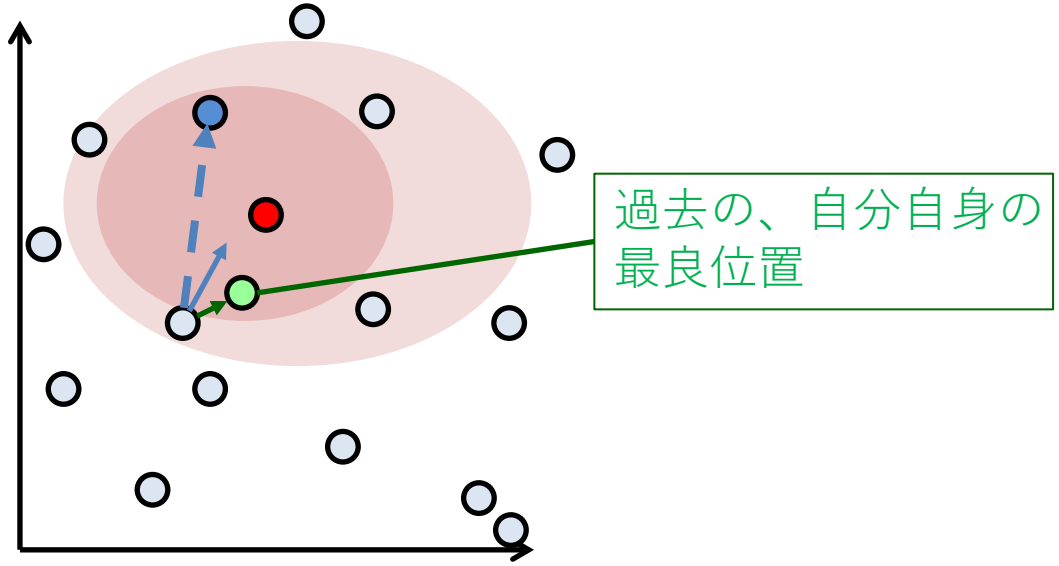
粒子群最適化法(4/8)

3. 最良解の適応度が十分でなければ探索を実施
   （それぞれのランダムな移動量と最良解の位置を考慮して決定）



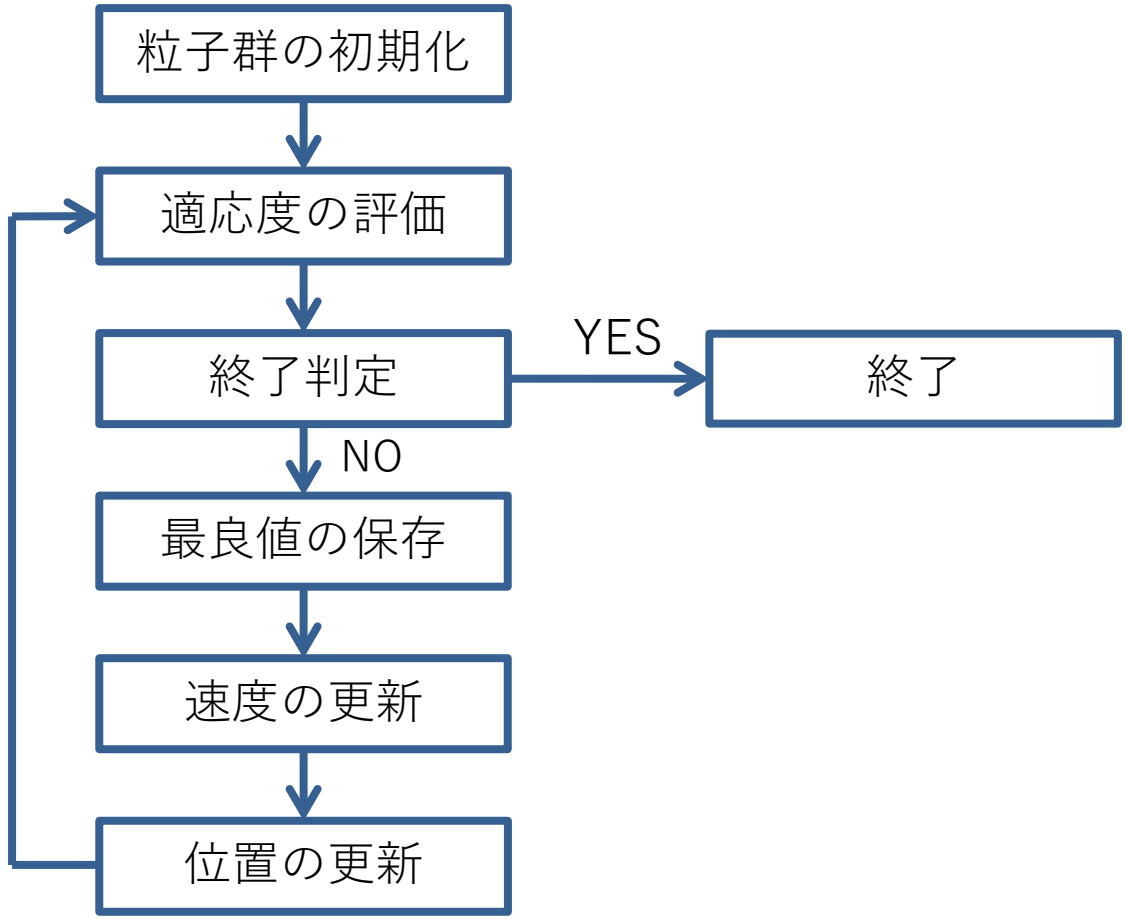
粒子群最適化法(5/8)

4. 2回目の探索以降は自分自身の最良位置なども考慮して
   移動量を決定



粒子群最適化法(6/8)

- 群（解集団）に含まれる複数個の探索個体が、群で情報を共有しながら最良値を探索



粒子群最適化法(7/8)

- 位置と速度を持った粒子群としてモデル化される

  位置 x：解の現在位置（値）、速度 v：解の移動量

  粒子群最適化法の更新式

  (1) $v = wv + c_1 r_1 (G_{best} - x) + c_2 r_2 (P_{best} - x)$

  (2) $x = x + v$

  w: 慣性定数
  v: 探索個体の速度
  x: 探索個体の位置
  c_1, c_2: 学習係数

  r_1, r_2: [0, 1]の一様乱数
  P_{best}: 各探索個体の最良位置
  G_{best}: 個体群全体での最良位置

- 各粒子が最良値に移動するため、収束性が高い

粒子群最適化法(8/8)

- 利点
  - 実装が容易
  - 各粒子が最良値に移動するため、収束性が高い
    （永遠に探索が続くことはほとんどない）

- 欠点
  - 初期解を十分にランダムに配置しなければ局所解に陥る
  - 解空間が離散である場合には使用できない

まとめ

- 組み合わせ最適化問題の概要を学んだ。
- 発見的方法により近似的な解を得る欲張り法について学んだ。
- 探索的方法により近似的な解を得る粒子群最適化法について学んだ。
- 組み合わせ最適化問題の具体的な事例として巡回セールスマン問題を取り上げその解法について学んだ。

復習問題

1. 決定変数が離散的で、全解を列挙するしかない問題の例を一つ挙げよ。

2. 問題の要素を複数の部分問題に分割し、その評価値の高いものを取り込んで解を得る発見的な近似解法を何というか？

3. 複数個の探索個体が、情報を共有しながら最良値見つける探索的な近似解法を何というか？

4. 粒子群最適化法では、探索個体の評価値として何を用いるか？

次回の講義

- 進化型計算
  - 遺伝的アルゴリズムの基礎
  - 個体の遺伝子型と表現型
  - 遺伝的アルゴリズム
  - 選択
  - 交叉（こうさ）
  - 突然変異
  - 遺伝子操作の実現方法