＜数理計画モデル＞

・線形計画問題

・ネットワーク計画問題

簡単に解ける

・非線形計画問題

・組合せ計画問題

難しい
（厳密な最適化は困難な場合が多い）

組合せ計画モデル

<生産計画問題＞

目的関数： $\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} \longrightarrow$ 最大

制約条件： $\mathrm{A}\boldsymbol{x} \leqq \boldsymbol{b}, \quad \boldsymbol{x} \geqq 0$

$\boldsymbol{x}$ の各要素は整数

整数計画問題

<固定費つき輸送問題>

2つの倉庫A_1, A_2から取引先B_1, B_2, B_3の注文を満たすように品物を送る。

注文量

B_1	20
B_2	30
B_3	15

輸送コスト

	B_1	B_2	B_3
A_1	5	3	2
A_2	2	7	4

固定費

A_1	300
A_2	200

z_i : 輸送費と固定費をあわせた各倉庫A_iの費用

$$z_1 = \begin{cases} 300 + 5x_{11} + 3x_{12} + 2x_{13} & \text{倉庫}A_1\text{を使用するとき} \\ 0 & \text{倉庫}A_1\text{を使用しないとき} \end{cases}$$

$$z_2 = \begin{cases} 200 + 2x_{21} + 7x_{22} + 4x_{23} & \text{倉庫}A_2\text{を使用するとき} \\ 0 & \text{倉庫}A_2\text{を使用しないとき} \end{cases}$$

$$y_i = \begin{cases} 1 & \text{倉庫}A_i\text{を使用するとき} \\ 0 & \text{倉庫}A_i\text{を使用しないとき} \end{cases}$$

$$x_{11} + x_{12} + x_{13} = \leqq 65\, y_1$$
$$x_{21} + x_{22} + x_{23} = \leqq 65\, y_2$$

$$z_1 = 300\, y_1 + 5x_{11} + 3\, x_{12} + 2\, x_{13}$$
$$z_2 = 200\, y_2 + 2x_{21} + 7\, x_{22} + 4\, x_{23}$$

$$x_{11} + x_{21} = 20$$
$$x_{12} + x_{22} = 30$$
$$x_{13} + x_{23} = 15$$

$z_1 + z_2$ の最小化

0-1 計画問題

＜ナップサック問題＞

n個の品物がある．品物iの重さ:$a_i$kg　利用価値:c_i とする．ナップサックには全部で$b$kgしか詰めない．利用価値の総計が最大となる品物を選ぶにはどうすればよいか．

目的関数： $\sum_{i=1}^{n} c_i x_i \longrightarrow$ 最大

制約条件： $\sum_{i=1}^{n} a_i x_i \leqq b$

$$x_i = 0,1 \quad (i=1,..,n)$$

$$x_i = \begin{cases} 1, & \text{品物}i\text{を選ぶとき} \\ 0, & \text{品物}i\text{を選ばないとき} \end{cases}$$

組合せ計画問題

有限個の要素からなる実行可能領域のなかで目的関数が最小となる解を見つける問題

例）・最短路問題

・線形計画問題

有限個の実行可能基底解から最適解を見つける

・ネットワーク計画問題

特殊な線形計画問題

＜組合せ計画問題＞

実行可能解の数は有限 → すべての実行可能解の目的関数を計算すれば最適解が求まる

n個の0-1変数x_i(i=1,..,n) の組

$\boldsymbol{x}=(x_1,\ldots, x_n)$

のとりうる値　2^n 個

変数が数十個程度の問題ですら現実には取り扱えない

計算の複雑さ

計算量 : アルゴリズムが停止するまでに実行
される演算の総数

大きさNの問題をf(N)回の演算で解くこと
ができる ⟶ 計算量O(f(N))

f(N)がNのべき乗で表される
⟶ 多項式時間アルゴリズム

f(N)が２^{N}やN！
⟶ 指数時間アルゴリズム

＜多項式時間アルゴリズム＞

大規模な問題に対しても効率的である

＜指数時間アルゴリズム＞

計算量が爆発的に増加

多項式時間アルゴリズムが存在する問題

→ クラスPに属する　　　多項式時間（polynomial time）

NP困難な問題：多項式時間アルゴリズムの存在が知られていない

非決定性計算による多項式時間（nondeterministic polynomial time ）

非決定性計算による多項式時間アルゴリズム
が存在する問題 → クラスNPに属する

＜帰着可能性＞

問題Aが多項式オーダーの計算で問題Bに変換できる

→ 問題Aは問題Bに帰着可能

＜NP困難(NP-hard)＞

NPの任意の問題Aが，ある一つの問題Bに帰着可能

→ BはNP困難(NP-hard)である

＜NP完全(NP-complete)＞

NP困難な問題BがさらにNPに属す場合

→ BはNP完全(NP-complete)である

＜クラスPに属する問題の例＞

・線形計画問題
・ネットワーク計画問題

＜NP完全問題の例＞

・充足可能性問題
・整数計画問題
・ナップサック問題
・巡回セールスマン問題
・スケジューリング問題
・集合分割問題

欲張り法

解を段階的に構築していく際に，常にその段階で最善と思われるものを取り入れていく方法

典型的な例：最小木問題に対するクラスカル法

一般に，このような単純な方法では問題の（大域的）最適解が構築できる保証はない

分かりやすく計算量も少ない

→ 近似解法の設計にしばしば採用される

分枝限定法

実行可能解を列挙するために場合分けを行っていく過程で，最適解が得られる見込みのない不必要な場合分けをできるだけ省略して，探索する範囲を絞り込むことにより計算時間を短縮する方法.

様々な問題に対して用いることのできる一般的な計算原理

例）ナップサック問題

目的関数：$\sum_{i=1}^{n} c_i x_i \longrightarrow$ 最大

制約条件：$\sum_{i=1}^{n} a_i x_i \leqq b$

$x_i = 0,1 \quad (i=1,..,n)$

ただし，c_i, a_i, bはすべて正の整数とし，

$$\frac{c_1}{a_1} \geqq \frac{c_2}{a_2} \geqq \cdots \geqq \frac{c_n}{a_n}$$

となるように，まえもって変数の添字iは並べ換えられていると仮定する．

目的関数： $7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow$ 最大

制約条件： $4x_1 + 5x_2 + x_3 + 3x_4 \leqq 6$

$$x_i = 0,1 \quad (i=1,..,4)$$

＜連続緩和問題＞

目的関数： $7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow$ 最大

制約条件： $4x_1 + 5x_2 + x_3 + 3x_4 \leqq 6$

$$0 \leqq x_i \leqq 1 \quad (i=1,..,4)$$

最適解： $x = (1,2/5,0,0)^{\mathrm{T}}$　元の問題の実数最適解

連続緩和問題は元の問題の目的関数値の上界値を与える.

連続緩和問題の実数最適解により，元の問題の近似最適解を容易に得ることができる.

＜分枝図＞

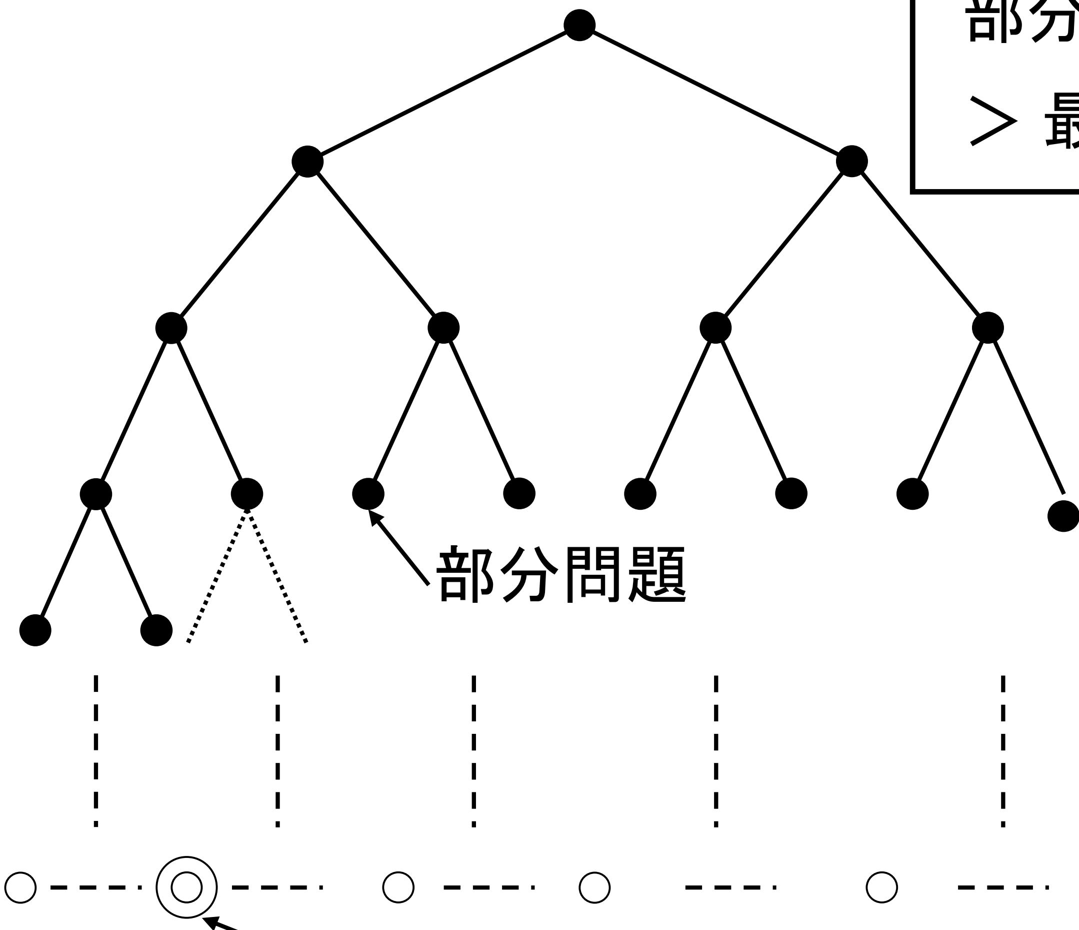
・最上部の節点：どの変数も固定されていない状態

・最下部の節点：すべての変数が0または1に固定された状態

・途中の節点：一部の変数の値が0または1に固定され，それ以外の変数は固定されていないような問題

→ 部分問題

分枝図（列挙木）



＜分枝限定法＞

部分問題の解の下界値
＞ 最適解の上界値（暫定解の値）

↓

その部分問題から最適解は得られないので探索打ち切り（限定操作）

A*アルゴリズム

実数最適解: $x = (1, 2/5, 0, 0)^T$ 　　z=10.2

目的関数: $7x_1 + 8x_2 + x_3 + 2x_4 \longrightarrow$ 最大

制約条件: $4x_1 + 5x_2 + x_3 + 3x_4 \leqq 6$

$x_i = 0,1 \quad (i=1,..,4)$

近似最適解: $x = (1, 0, 1, 0)^T$

暫定解: $x = (1, 0, 1, 0)^T$ 　　暫定値: $z^* = 8$

部分問題に対して，その最適解がもとの問題の最適解を与える可能性があるかどうかテスト

可能性がないことが判明すれば，その部分問題を解くことをやめる

部分問題の最適解が得られる

部分問題には実行可能解が存在しないことが判明

部分問題の終端

(a) $x_1 = 0$ に固定した部分問題

目的関数: $8x_2 + x_3 + 2x_4 \longrightarrow$ 最大

制約条件: $5x_2 + x_3 + 3x_4 \leqq 6$

$x_i = 0,1 \quad (i=2,..,4)$

＜連続緩和問題＞

目的関数: $8x_2 + x_3 + 2x_4 \longrightarrow$ 最大

制約条件: $5x_2 + x_3 + 3x_4 \leqq 6$

$0 \leqq x_i \leqq 1 \quad (i=2,..,4)$

実数最適解: $(x_2, x_3, x_4) = (1, 1, 0)^T$ z= 9

部分問題の最適解 $x = (0,1, 1, 0)^T$ が得られた

$\longrightarrow$ 終端できる

(b) $x_1=1$,　$x_2=0$ に固定した部分問題

目的関数:　$7+x_3+2x_4$ ⟶ 最大

制約条件:　$4+x_3+3x_4\leqq 6$

$x_i=0,1\quad(i=3,4)$

＜連続緩和問題＞

目的関数:　$7+x_3+2x_4$ ⟶ 最大

制約条件:　$4+x_3+3x_4\leqq 6$

$0\leqq x_i\leqq 1\quad(i=3,4)$

実数最適解: $(x_3, x_4)=(1, 1/3)^T$　　$z\fallingdotseq 8.67$

⟶　終端できない

目的関数値が暫定値$z^*=8$より大きい

(c) $x_1 = 1$,　$x_2 = 1$ に固定した部分問題

目的関数：　$15 + x_3 + 2x_4$ ⟶ 最大

制約条件：　$9 + x_3 + 3x_4 \leqq 6$

$x_i = 0,1 \quad (i=3,4)$

明らかに実行可能解をもたない

⟶　終端できる

暫定解：それまでに得られている最良の実行可能解

暫定値：暫定解の目的関数値

＜部分問題のテストと終端＞

・部分問題の連続緩和問題の最適解の値（上界値）が暫定値より小さい

・部分問題の連続緩和問題の最適解が0-1条件を満たす

・部分問題が実行可能解をもたないことが判明

限定操作

＜分枝限定法＞

(0) 適当な方法で近似最適解を求め，それを暫定解，その目的関数の値を暫定値z^*とする．もとの問題P_0からいくつかの部分問題$P_1,..,P_m$を生成し、$A:=\{P_1,..,P_m\}$とおく.

(1) 集合Aから部分問題P_iを一つ選ぶ.

(1-1) 部分問題P_iが実行可能解をもたなければ終端.
$A:=A-\{P_i\}$としてステップ(3)へ.

(1-2) 部分問題P_iの最適解が得られ，$z_i \leqq z^*$ならば終端.
$z_i > z^*$ ならば$z^* := z_i$とし暫定解を更新し終端.
$A:=A-\{P_i\}$としてステップ(3)へ.

(1-3) 部分問題P_iの上界値$\overline{z_i}$が得られ，$\overline{z_i} \leqq z^*$ならば終端.
$A:=A-\{P_i\}$としてステップ(3)へ. $\overline{z_i} > z^*$ ならばステップ(2)へ.

(2) 部分問題P_iからいくつかの部分問題$P_j,..,P_k$を生成し，
$A := A \cup \{P_j,..,P_k\} - \{P_i\}$とおく．ステップ(1)へ戻る．

(3) $A=\varphi$ならば計算終了（暫定解はもとの問題P_0の最適解）．さもなければステップ(1)へ戻る．

分枝限定法の適用

(0) 近似最適解を暫定解とする。

暫定解: $x=(1, 0, 1, 0)^T$　　暫定値: $z^*=8$

(1) x_1＝0 に固定した部分問題を解く

(a)より部分問題の最適解 $x=(0,1, 1, 0)^T$が得られたから終端できる

目的関数値$z=9$ は暫定値$z^*=8$より大きいから、これを新たな暫定解とし、暫定値$z^*=9$とする。

(2) $x_1=1$ に固定した部分問題を解く

目的関数： $7+8x_2+x_3+2x_4 \longrightarrow$ 最大

制約条件： $4+5x_2+x_3+3x_4 \leqq 6$

$x_i=0,1 \quad (i=2,..,4)$

＜連続緩和問題＞

目的関数： $7+8x_2+x_3+2x_4 \longrightarrow$ 最大

制約条件： $4+5x_2+x_3+3x_4 \leqq 6$

$0 \leqq x_i \leqq 1 \quad (i=2,..,4)$

実数最適解： $(x_2, x_3, x_4)=(2/5, 0, 0)^T$

目的関数値z=10.2 は暫定値z*=9より大きいから、最適解を含む可能性がある。そこで新たな部分問題を生成する。

(3) $x_1=1,\ x_2=0$ に固定した部分問題を解く

(b)より部分問題の実数最適解：$(x_3, x_4)=(1, 1/3)^T$
$z \fallingdotseq 8.67$ が得られる。しかし$z=14/3$は暫定値$z^*=9$
より小さい。
したがって、この部分問題は終端する。

(4) $x_1=1,\quad x_2=1$ に固定した部分問題を解く

(c)よりこの部分問題は実行可能解をもたない。
したがって、この部分問題は終端する。

(5) すべての部分問題が終端したので、暫定解

$x=(0,1, 1, 0)^T$ が最適解となる。

近似解法

問題の大きさに関する多項式時間で解ける問題

$\longrightarrow$ クラスPに属する

ある問題を場合分けによって解くとき，それぞれの場合に対する計算が問題の大きさに関する多項式時間で実行できる.

$\longrightarrow$ クラスNPに属する

$P \subseteq NP$

現実の応用に現れる組合せ最適化問題はたいていクラスNPに属している

↓

NP困難の問題に対しては，厳密な最適解を求めることはあきらめ，良い近似最適解が比較的短時間で得られればそれで満足しようという考え方が一般に受け入れられている．

ヒューリスティック解法（発見的解法）

<巡回セールスマン問題>

n個の節点をもつネットワーク G=(V,E) において各枝 $(i,j)\in$E の長さ a_{ij}があたえられたとき，すべての節点をちょうど一度ずつ訪問してもとに戻る巡回路のなかで最短のものを見つける問題.

欲張り法に基づくヒューリスティック解法

<最近近傍法>

ある節点から出発して，まだ訪問していない隣接節点のなかで最も近い節点へ移動していく方法

組合せ問題の難しさ
－ハミルトン経路問題－

セールスマンが全ての都市を1回ずつ通過して、出発地に戻って来る経路で最も短いものを捜す問題です。

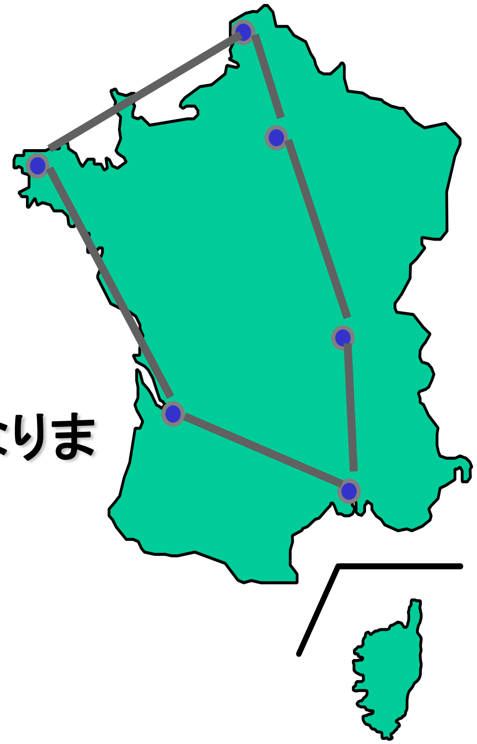
・6都市ならば、
　　5!/2= ５×４×３×２ ／ ２ =60通り

・n都市ならば、
　　（nー１）！/2通り

近年, 新聞や科学雑誌でも取り上げられて有名になりました。

TSP（Traveling Salesman Problem）

原型：ハミルトン経路問題



東京大学工学部計数工学科　松井知己氏資料から

＜計算時間の実感＞

例えば、1MIPS (mega instructions per second)の計算機では、1秒間に100万回の計算ができます。つまり、1step に 10^{-6}秒かかりますが、nが大きくなると、以下のような計算時間になり、n！通りの大きさが実感できて、全てのパターンを計算し、その結果を元に最も良い解を導出することが不可能であることが分かります。

n	10	100	1,000	10,000
n	10^{-5}秒	10^{-4}秒	0. 001秒	0. 01秒
n^2	10^{-4}秒	0. 01秒	1秒	100秒
n^3	0. 001秒	1秒	16. 6分	277時間
2^n	0. 001秒	10^{14}世紀	10^{284}世紀	？∞？
n！	0. 036秒	10^{141}世紀	10^{2551}世紀	？∞？

＜挿入法＞

(0) 節点i_0を任意に選び，R:={i_0}とする.

(1) 巡回路Rとの距離$d(R, i)$が最小となる節点$i \notin R$を選ぶ.

(2) $a_{ij} = d(R, i)$を満たす節点$j \in R$に対して，巡回路から節点jを端点とする一つの枝(j, k)を取り除き，かわりに枝(i, j)と(i, k)を付け加える．新しい巡回路を$R := R \cup \{i\}$とする.

(3) すべての節点を通る巡回路が得られたなら計算終了．さもなければステップ(1)へ戻る.

ただし，$$d(R, i) = \min_{j \in R} \{a_{ij}\}$$

局所探索法とメタヒューリスティクス

＜xの近傍＞

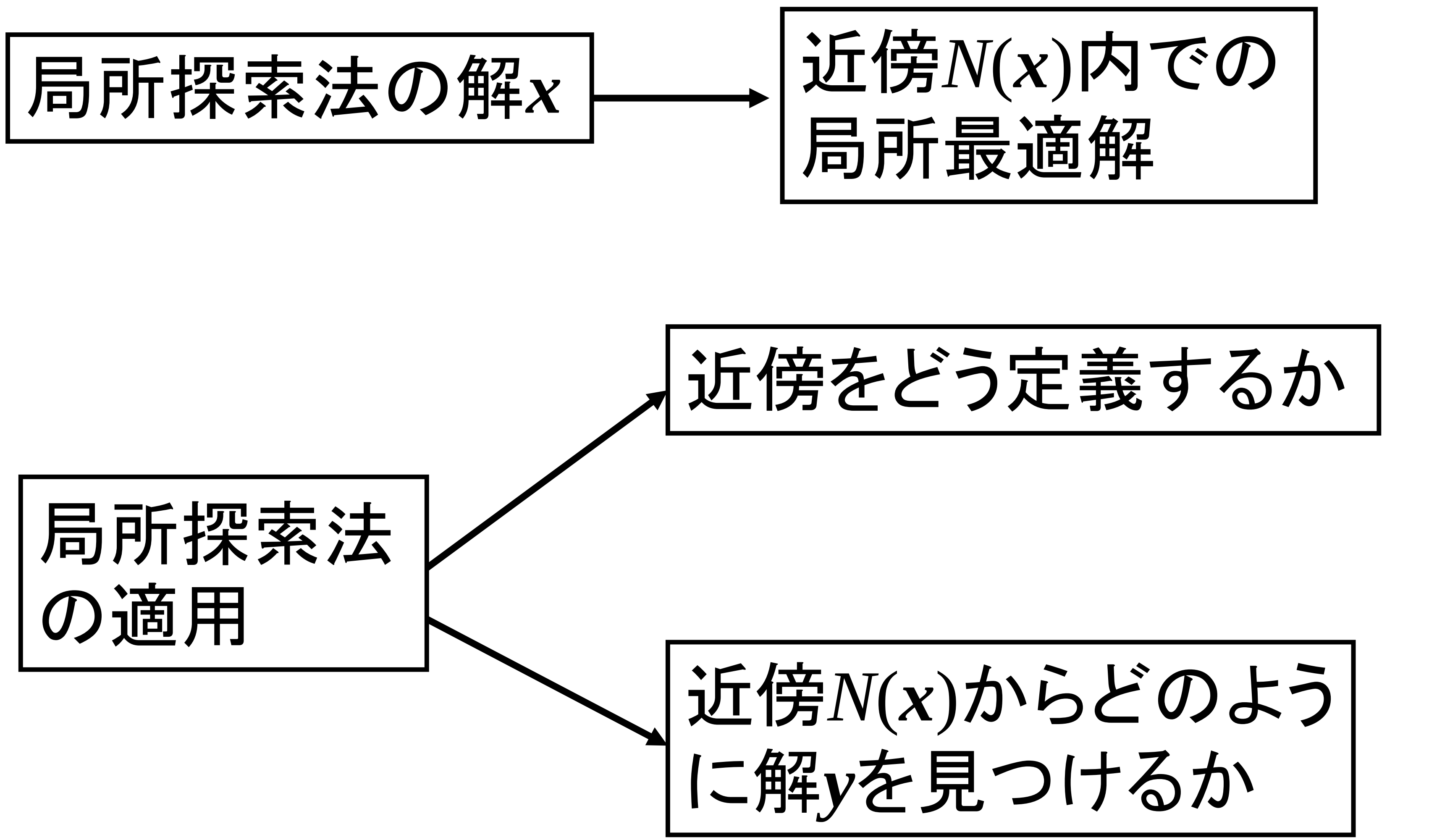
$N(\boldsymbol{x})$: 任意の実行可能解 $\boldsymbol{x}$ に対して，その一部分を修正して得られる解の集合

＜局所探索法＞

(0) 初期解xを選ぶ.

(1) 現在の解 x の近傍 $N(\boldsymbol{x})$ から $f(\boldsymbol{x}) < f(\boldsymbol{y})$ を満たす解 y を選ぶ. そのような解 y が$N(\boldsymbol{x})$内に存在しなければ計算終了.

(2) x を y で置き換えてステップ(1)へ戻る.


局所探索法の解x
近傍$N(x)$内での局所最適解
局所探索法の適用
近傍をどう定義するか
近傍$N(x)$からどのように解yを見つけるか

例）巡回セールスマン問題

巡回路xの近傍$N(\boldsymbol{x})$

隣り合わない２本の枝を巡回路xから取り去り，別の２本の枝を付け加えて得られる巡回路の全体.

⟶ **２-opt近傍**

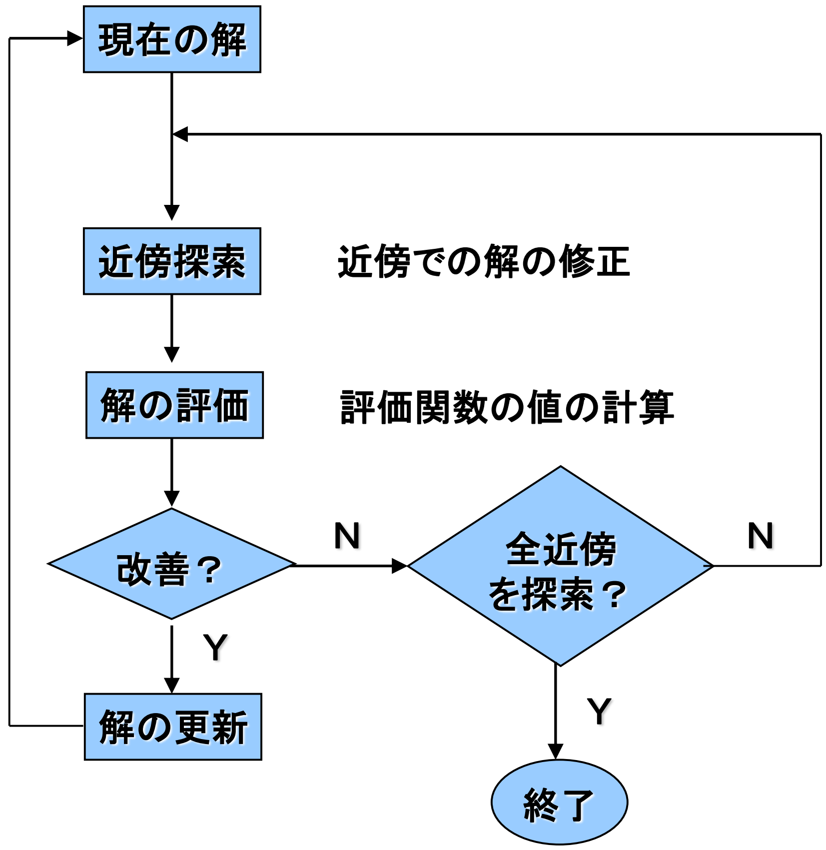
＜k-opt近傍＞

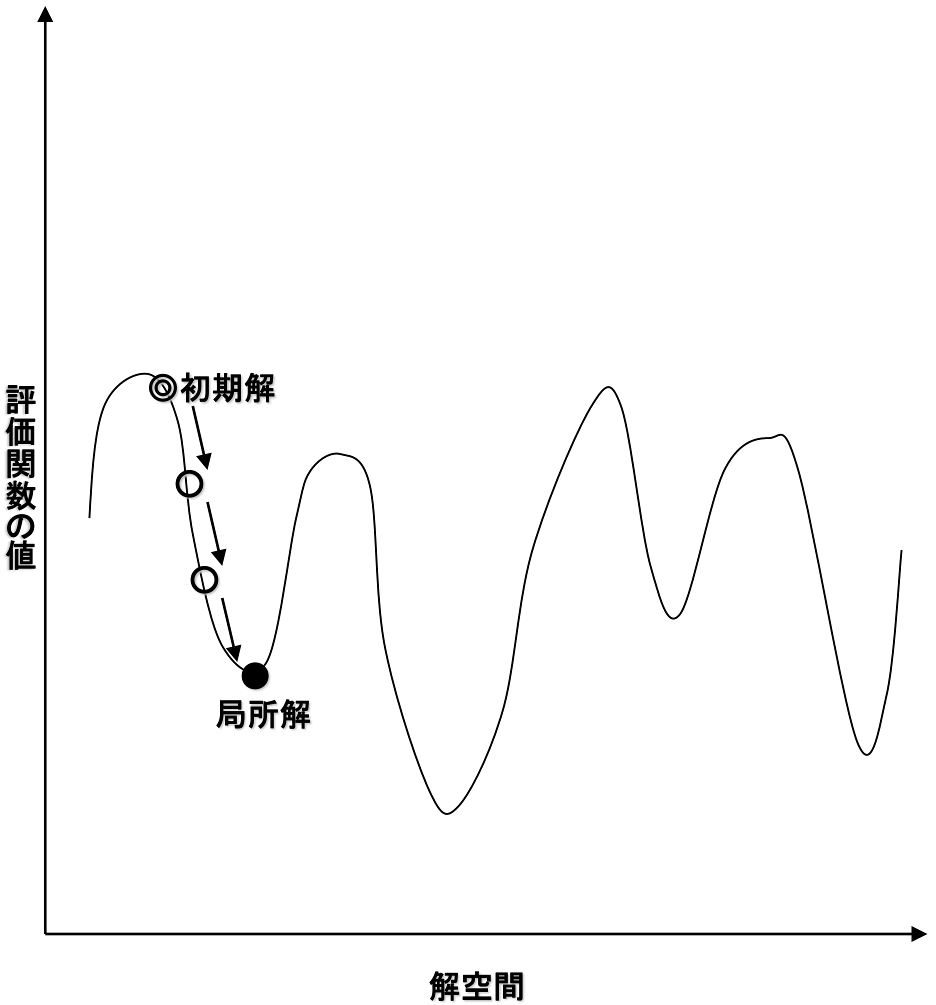
隣り合わない k本の枝を巡回路xから取り去り，別の k本の枝を付け加えて得られる巡回路の全体.

近傍内の解の探索：$O(n^k)$の計算量

Lin-Kernighanのアルゴリズム：2-optと3-optを巧妙に組み合わせる

局所探索法（山登り法）





山登り法（局所探索法）

＜局所探索法＞

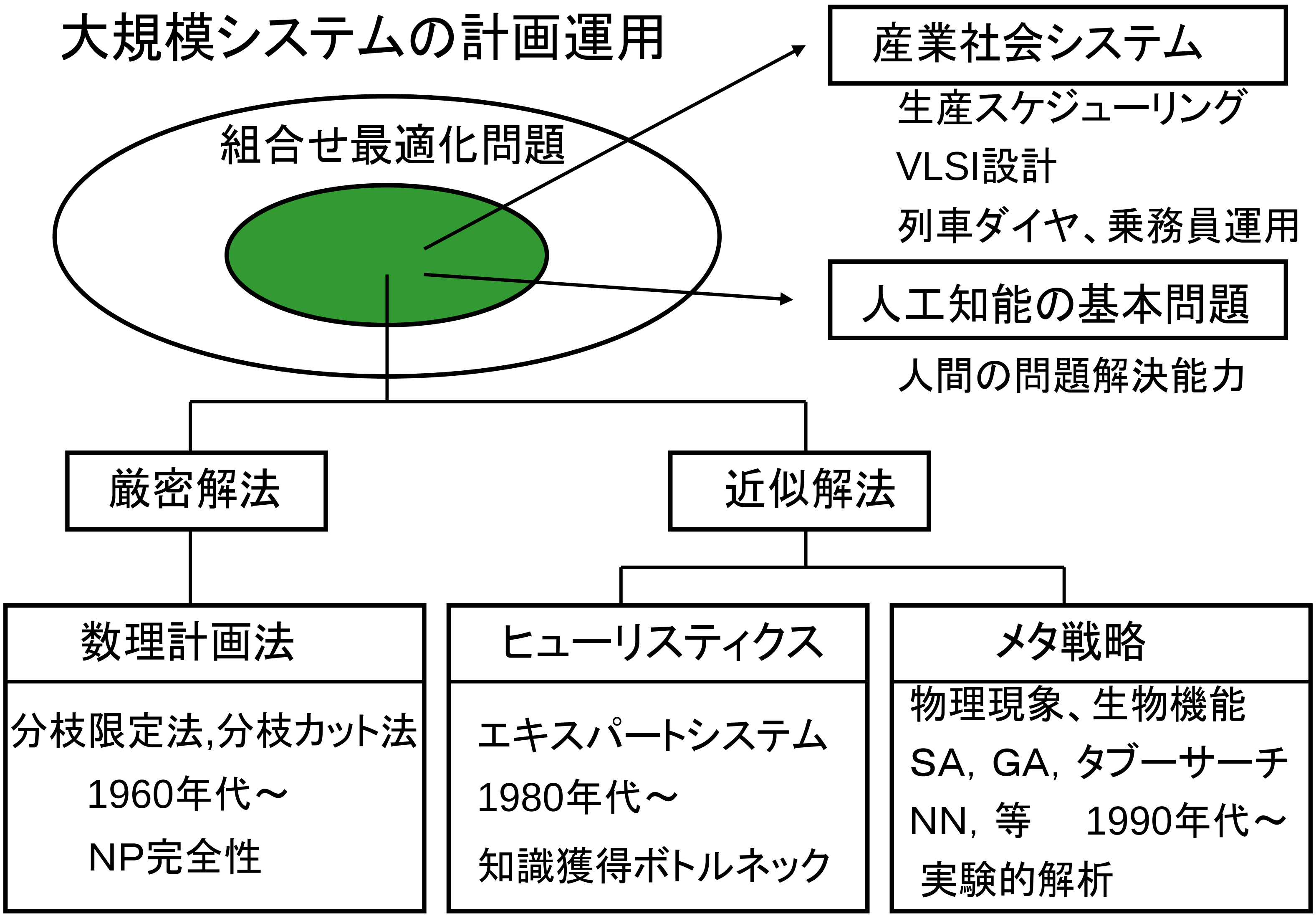
いったん局所最適解に到達すれば，そこから抜け出すことはできない

得られた局所最適解が大域的最適解である保証はない

＜メタヒューリスティクス＞

改悪となるような解への移動も許すことによって，好ましくない局所最適解に補足されることを避けようとする方法

＜大規模組合せ最適化問題の解法＞



メタヒューリスティクスの方式

＜山登り法とメタヒューリスティクス＞

基本的な探索方法は、通常「山登り法」と呼ばれる局所探索法です。

これは、初期解からスタートして、解を徐々に改善していく手法で、局所解に到達したら探索を終了するものです。

これに対して、局所解から脱出して、さらに最適に近い解を効率良く探索しようとする手法が幾つか提案されいます。メタヒューリスティクスあるいは、メタ戦略と呼ばれるもので、遺伝アルゴリズム（GA法）、シミュレーテッド・アニーリング（SA法）、タブーサーチ（TA）等があります。

＜遺伝アルゴリズム（GA法）＞

生物の集団が自然淘汰により進化していく過程を模したものです。

複数の解（集団）を用意し、それらを組み合わせることにより、より良い解（進化）を求めていこうという手法です。

＜シミュレーテッド・アニーリング（SA法）＞

焼きなまし法と呼ばれもので、温度を下げることにより、より強固な固体結晶を得ようとする物理過程（熱力学）をもしたものです。例えば、刀鍛冶が鉄を熱しては水で冷却する作業を繰り返す（焼きなまし）ことにより、切れ味の良い（強固な）刀を作る過程です。

最も強固な状態（最適解）に至るには、単に一度に冷却したのでは駄目で、何度も加熱（解の改悪）と冷却（解の改善）を繰り返す必要があります。

＜タブーサーチ（TA）＞

人間には記憶があり、学習により最適解にたどり着くことができます。山登りに例えれば、一度通過した頂上（局所解）に逆戻りすることを禁じる（ターブーとする）ことにより、効率的に真の最高峰に到達できます。

＜焼きなまし法(シミュレーティッド･アニーリング法)＞

統計物理の分野で研究されてきた焼きなまし過程の考え方に基づいて開発された組合せ最適化問題に対する方法

現在の解の近傍からランダムに解を選ぶ

改良ならば新しい解とする

改悪の場合でも目的関数の差に応じた確率で新しい解とする

＜特徴＞

改悪となる解を採用する確率を温度と呼ばれるパラメータを用いて変化させる

＜焼きなまし法＞

(0) 凍結温度T_{freeze}>0を定め，初期温度$T>T_{freeze}$と初期解 $\boldsymbol{x}$ を選ぶ.

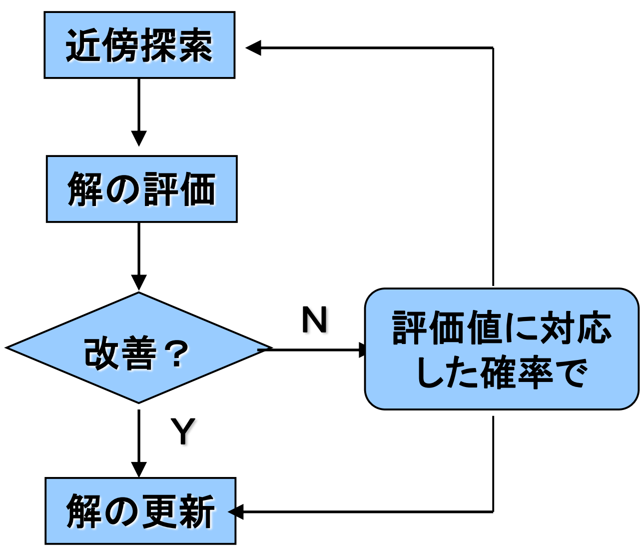
(1) 現在の解$\boldsymbol{x}$の近傍N($\boldsymbol{x}$)からランダムに解$\boldsymbol{y}$ を選び，$\Delta := f(\boldsymbol{y}) - f(\boldsymbol{x})$ とおく.

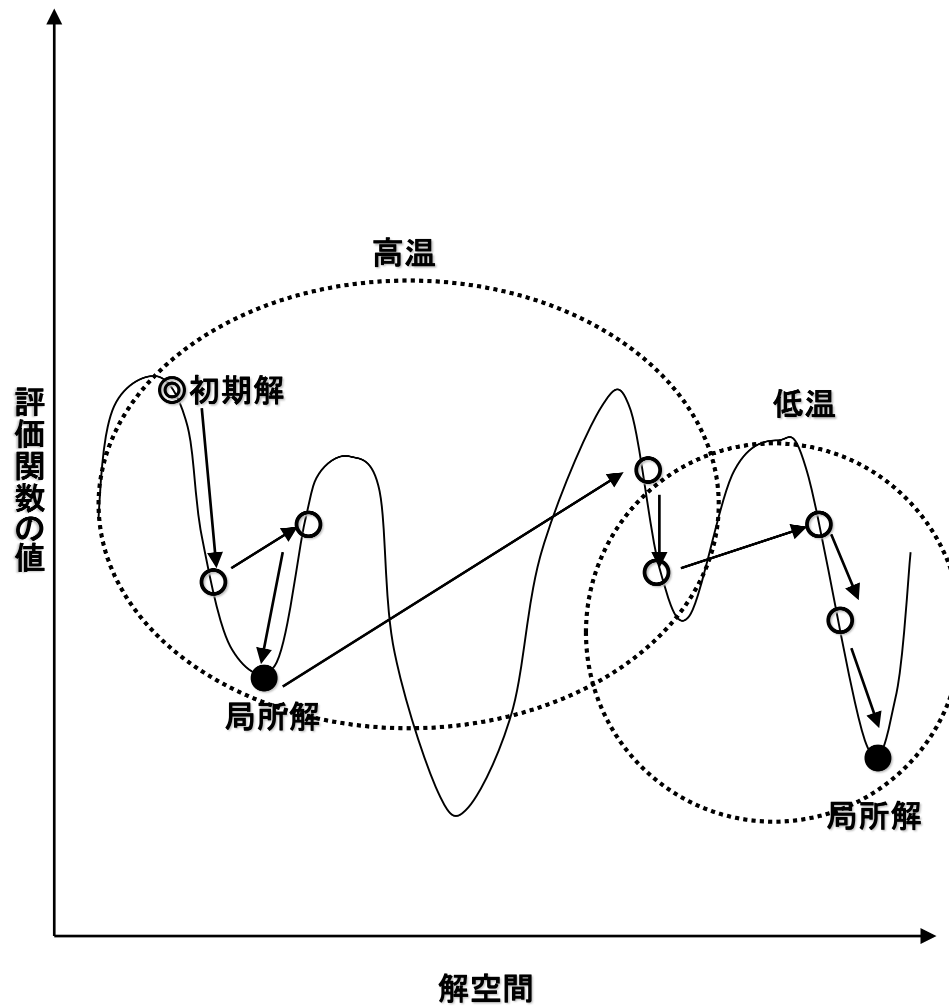
(2) $\Delta<0$ならば $\boldsymbol{x}$ を $\boldsymbol{y}$ で置き換える.

(3) $\Delta \geqq 0$ならば，区間[0,1]から実数ξをランダムに選び，$\xi < e^{-\Delta/T}$であれば $\boldsymbol{x}$ を $\boldsymbol{y}$ で置き換える.

(4) $T \leqq T_{freeze}$ が満たされれば計算終了．そうでなければ，新しい温度$T_{new} \leqq T$を定める. TをT_{new}で置き換えて，ステップ (1)へ戻る.

＜SA法＞





シミュレーテッド・アニーリング法（SA法）

＜タブー探索法＞

局所的最適解に到達した後，さらに探索を継続する.

↓

近傍N(x)においてx自身を除く最良の解yに移動する

↓

過去の探索における移動の経験を蓄積しておき，同じ解の再探索を避ける.

過去の探索で現れた解や移動のパターンをタブーリストと呼ばれる集合の形で記憶しておき、そのリストに含まれない解のなかで最良のものに移動する方法

＜タブー探索法＞

(0) 初期解 $\boldsymbol{x}$ を選ぶ．タブーリストの最大長 l_{max} を定め，初期タブーリストを L_{tabu}:=φ とする．

(1) 現在の解 $\boldsymbol{x}$ の近傍 N($\boldsymbol{x}$) において $\boldsymbol{x}$ 自身とタブーリスト L_{tabu} に含まれる解を除く最良の解 $\boldsymbol{y}$ を見つけ，$\boldsymbol{x}$ を $\boldsymbol{y}$ で置き換える．

(2) タブーリスト L_{tabu} に新しい解 $\boldsymbol{x}$ を追加する．もしも L_{tabu} の大きさが l_{max} を越えれば，最も古い要素を L_{tabu} から取り除く．

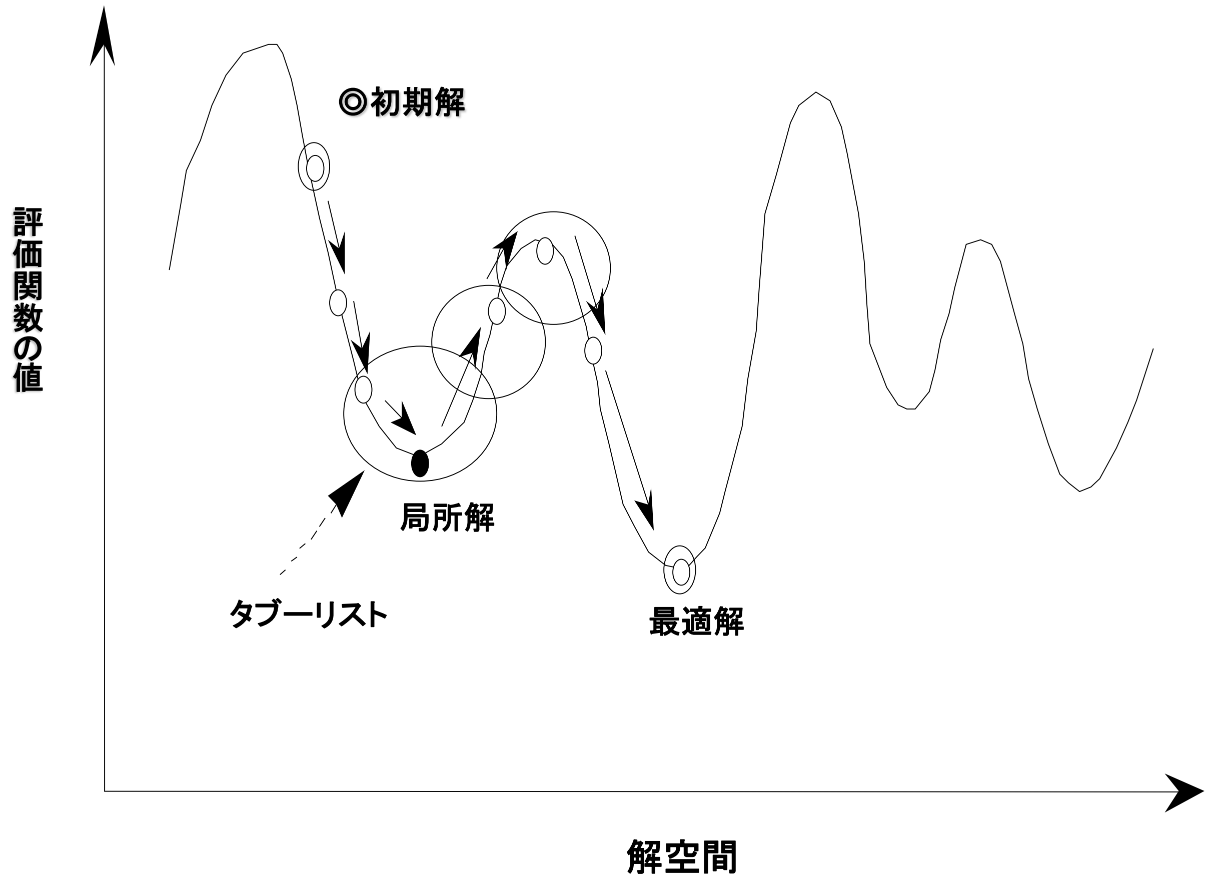
(3) 停止条件が満たされれば計算終了．そうでなければ，ステップ (1)へ戻る．

＜タブー探索法＞

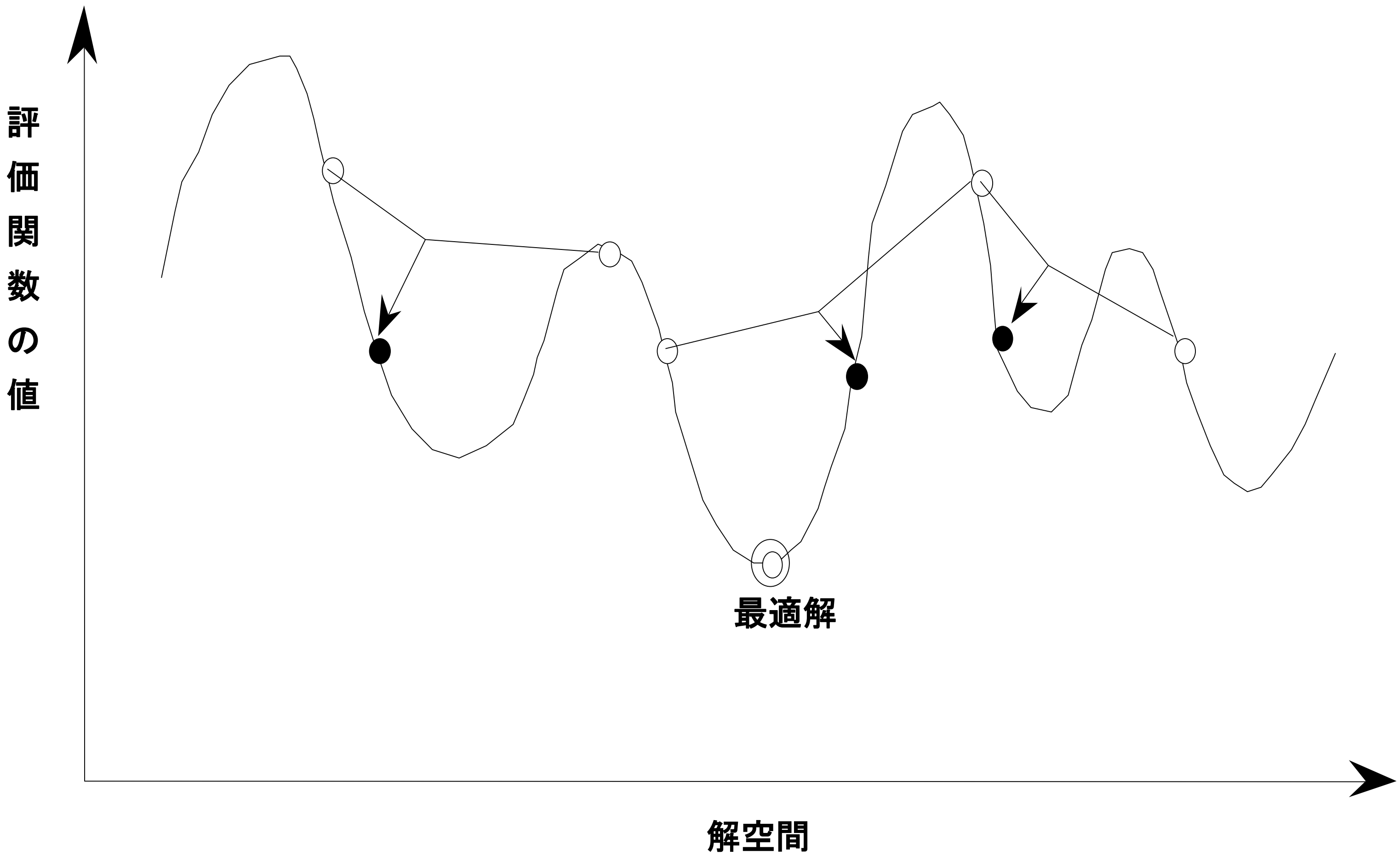
局所探索

↓

局所解を一定期間
探索禁止とする
（タブーリスト）



タブー探索法（TS法）



遺伝アルゴリズム（GA法）