メタヒューリスティクス

◆ヒューリスティクス

- 必ずしも最適解が得られるわけではないが，多くの場合，ある程度のレベルの近似解が得られる手法
  … 発見的手法，ヒューリスティック解法
- 欲張り法など

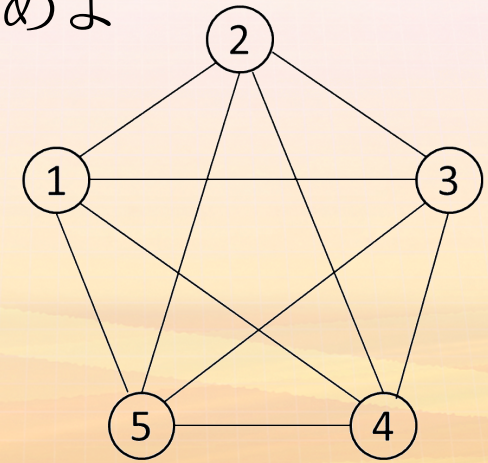
メタヒューリスティクス

- 局所最適解に捕捉されることを防ぐための
  発見的手法の枠組み群で，特定の問題に，
  限定されず汎用的に適用できる
- 代表的な枠組みに，局所探索法，タブー探索法，
  遺伝的アルゴリズム，焼きなまし法，
  ニューラルネットワーク，蟻コロニー最適化，
  粒子群最適化，…

メタヒューリスティクス

◆巡回セールスマン問題 (TSP)

- 点は都市，表の数字は都市間の距離を表す
- セールスマンがこれら5つの都市を1回ずつ通って巡回する時の最短経路を求めよ



	1	2	3	4	5
1	0	2	4	4	3
2	2	0	5	3	3
3	4	5	0	3	1
4	4	3	3	0	3
5	3	3	1	3	0

メタヒューリスティクス

◆巡回セールスマン問題 (TSP)

- 0-1整数最適化問題として定式化
- 都市の集合をV，都市の数をn
- 都市iから都市jが巡回路に含まれるとき　$x_{ij} = 1$,
  含まれないとき$x_{ij} = 0$
- 都市iと都市jの間の距離をc_{ij}
- 総移動距離zは

$$z = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

メタヒューリスティクス

◆巡回セールスマン問題 (TSP)

- 各都市には1回だけ他の1都市から入る

$$\sum_{i \in V} x_{ij} = 1 \quad (j = 1, 2, \cdots, n)$$

- 各都市からは1回だけ別の1都市に出ていく

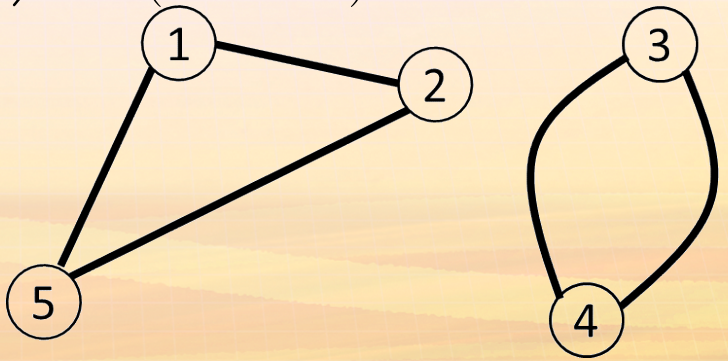
$$\sum_{j \in V} x_{ij} = 1 \quad (i = 1, 2, \cdots, n)$$

メタヒューリスティクス

◆巡回セールスマン問題 (TSP)

- 一部の都市だけで巡回路ができてはいけない
- 巡回路ができない … 枝の数が点の数より少ない
- TSPはすべての都市を通る巡回路を求める問題
- Vのすべての真部分集合$S \subsetneq V$ $(S \subset V)$に関して

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1$$



メタヒューリスティクス

◆巡回セールスマン問題 (TSP)

$$\text{最小化} \quad z = \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

$$\text{制約条件} \quad \sum_{i \in V} x_{ij} = 1 \quad (j \in V)$$

$$\sum_{j \in V} x_{ij} = 1 \quad (i \in V)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \quad (S \subsetneq V)$$

$$x_{ij} \in \{0, 1\}, \quad (i, j \in V)$$

メタヒューリスティクス

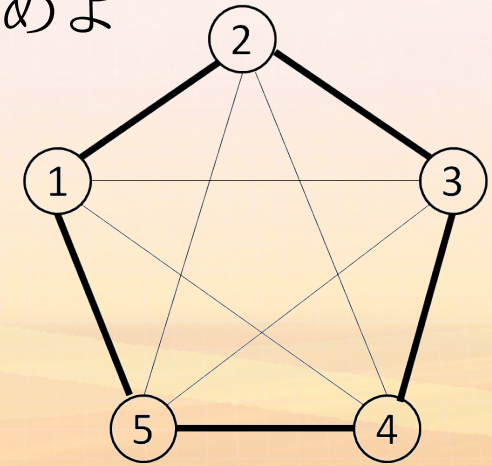
◆巡回セールスマン問題 (TSP)

- n都市のTSPは$(n-1)!/2$通りの巡回路が存在
- すべての巡回路を調べるのは不可能
- 欲張り法では最適解が得られない
- 代表的な難しい問題

メタヒューリスティクス

◆巡回セールスマン問題 (TSP)

- 点は都市，表の数字は都市間の距離を表す
- セールスマンがこれら5つの都市を1回ずつ通って巡回する時の最短経路を求めよ



	1	2	3	4	5
1	0	2	4	4	3
2	2	0	5	3	3
3	4	5	0	3	1
4	4	3	3	0	3
5	3	3	1	3	0

メタヒューリスティクス

◆巡回セールスマン問題 (TSP)

- n都市のTSPは$(n-1)!/2$通りの巡回路が存在
- すべての巡回路を調べるのは不可能
- 欲張り法では最適解が得られない
- 代表的な難しい問題

局所探索法

局所探索法

- 目的関数 $f(\boldsymbol{x})$ を最小化
- 任意の実行可能解 $\boldsymbol{x}$ の一部分を変更して得られる解の集合 $N(\boldsymbol{x})$ $\cdots$ $\boldsymbol{x}$ の近傍
- $f(x_1, x_2, x_3) \quad x_1, x_2, x_3 \in \{0, 1\}$

$$N_1(1, 0, 1) = \{(1, 0, 0), (1, 1, 1), (0, 0, 1)\}$$

局所探索法

(0) 初期解$\boldsymbol{x}$を選ぶ

(1) $\boldsymbol{x}$の近傍$N(\boldsymbol{x})$の中に$f(\boldsymbol{x}') < f(\boldsymbol{x})$を満たす実行可能解$\boldsymbol{x}'$を選ぶ
そのような実行可能解$\boldsymbol{x}'$が存在しなければ,
$\boldsymbol{x}$を解として出力して終了

(2) $\boldsymbol{x} \leftarrow \boldsymbol{x}'$として(1)へ

局所探索法

- 近傍を大きくとると，より良い解が見つかる可能性が大きくなるが，探索に要する計算量も大きくなる
- $f(\boldsymbol{x}') < f(\boldsymbol{x})$ を満たす $\boldsymbol{x}'$ が見つかった時点で直ちにそれを $\boldsymbol{x}$ とすれば，近傍内での最適解を見つけられない可能性
- $f(\boldsymbol{x}')$ を最小にする $\boldsymbol{x}'$ を選ぶのは計算量を要する

局所探索法

◆ナップサック問題への適用

- n個の品物
- 品物iは重さa_iで価値がc_i
- ナップサックに最大で重さbまで詰められる
- 価値の合計が最大になるには，どの品物を詰めるべきか

局所探索法

◆ナップサック問題への適用

- 品物iをナップサックに詰める時は　$x_i = 1$
  詰めない時は$x_i = 0$
- 解 $\boldsymbol{x} = (x_1, x_2, x_3, x_4) = (1, 1, 1, 0)$ が得られている
- $\boldsymbol{x}$の1成分の0と1を反転させた解の集合$N_1(\boldsymbol{x})$

$$N_1(1,1,1,0) = \{(0,1,1,0), (1,0,1,0), (1,1,0,0), (1,1,1,1)\}$$

局所探索法

◆ナップサック問題への適用

- 品物iをナップサックに詰める時は　$x_i = 1$
  詰めない時は$x_i = 0$
- 解 $\boldsymbol{x} = (x_1, x_2, x_3, x_4) = (1, 1, 1, 0)$ が得られている
- $\boldsymbol{x}$の1成分の0と1を反転させた解の集合$N_2(\boldsymbol{x})$

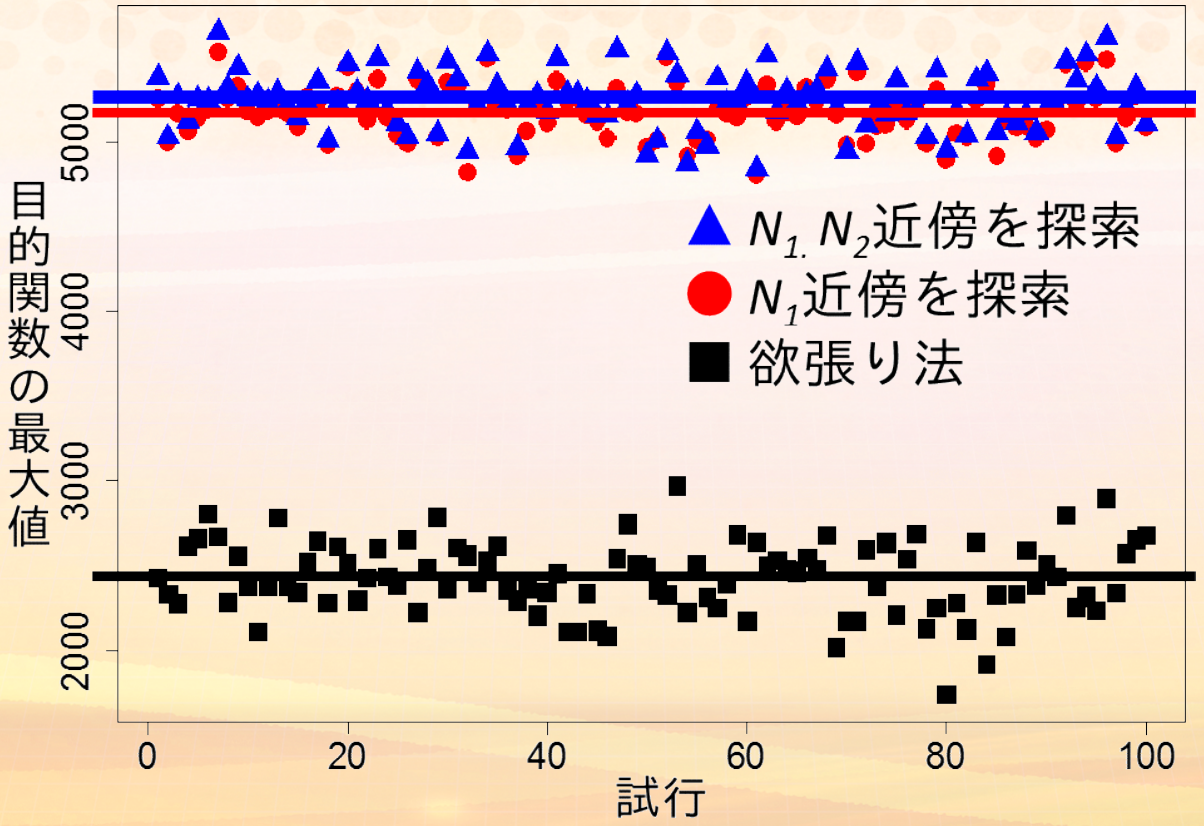
$$N_2(1,1,1,0) = \{(0,0,1,0), (0,1,0,0), (0,1,1,1), (1,0,0,0), (1,0,1,1), (1,1,0,1)\}$$

局所探索法

◆ナップサック問題への適用

- 100変数ナップサック問題100問

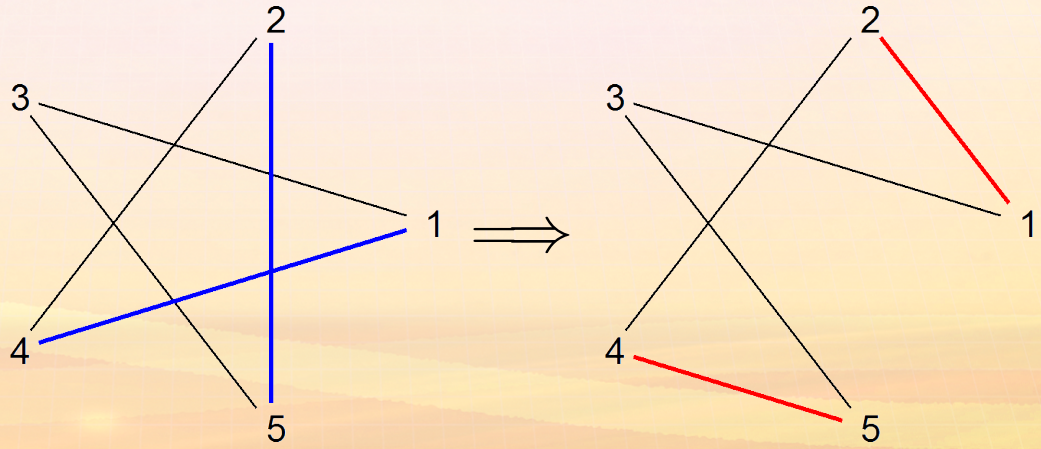
1. 欲張り法
2. 欲張り法 + N_1近傍探索
3. 欲張り法 + $N_1 \cup N_2$近傍探索


目的関数の最大値
2000
3000
4000
5000
0
20
40
60
80
100
試行
N_1, N_2近傍を探索
N_1近傍を探索
欲張り法

局所探索法

◆TSPへの適用

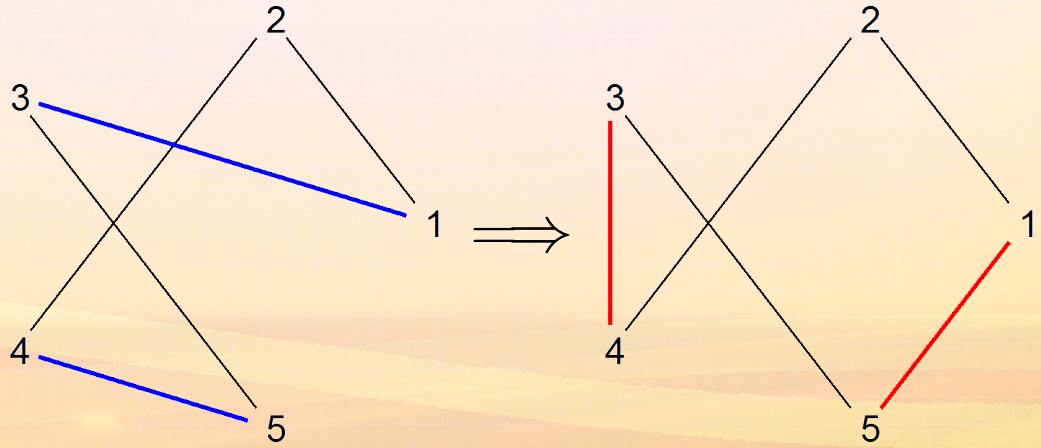
- 巡回路$\boldsymbol{x}$から隣り合わない2本の枝を取り除き，別の2本の枝を加えて得られる巡回路$\boldsymbol{x}'$の集合$N(\boldsymbol{x})$
  … **2-opt近傍**



局所探索法

◆TSPへの適用

- 巡回路$\boldsymbol{x}$から隣り合わない2本の枝を取り除き，別の2本の枝を加えて得られる巡回路$\boldsymbol{x}'$の集合$N(\boldsymbol{x})$
  … **2-opt近傍**



局所探索法

◆TSPへの適用

- 2-optでは，2本の枝を取り除いた場合，
  加える2本の枝は一意に決まる
- 都市数がnのTSPにおいて，1つの巡回路に対して
  2本の枝の取り除き方は約${}_nC_2$通り … $O(n^2)$存在
- 3-opt近傍等に比べると近傍が小さい

局所探索法

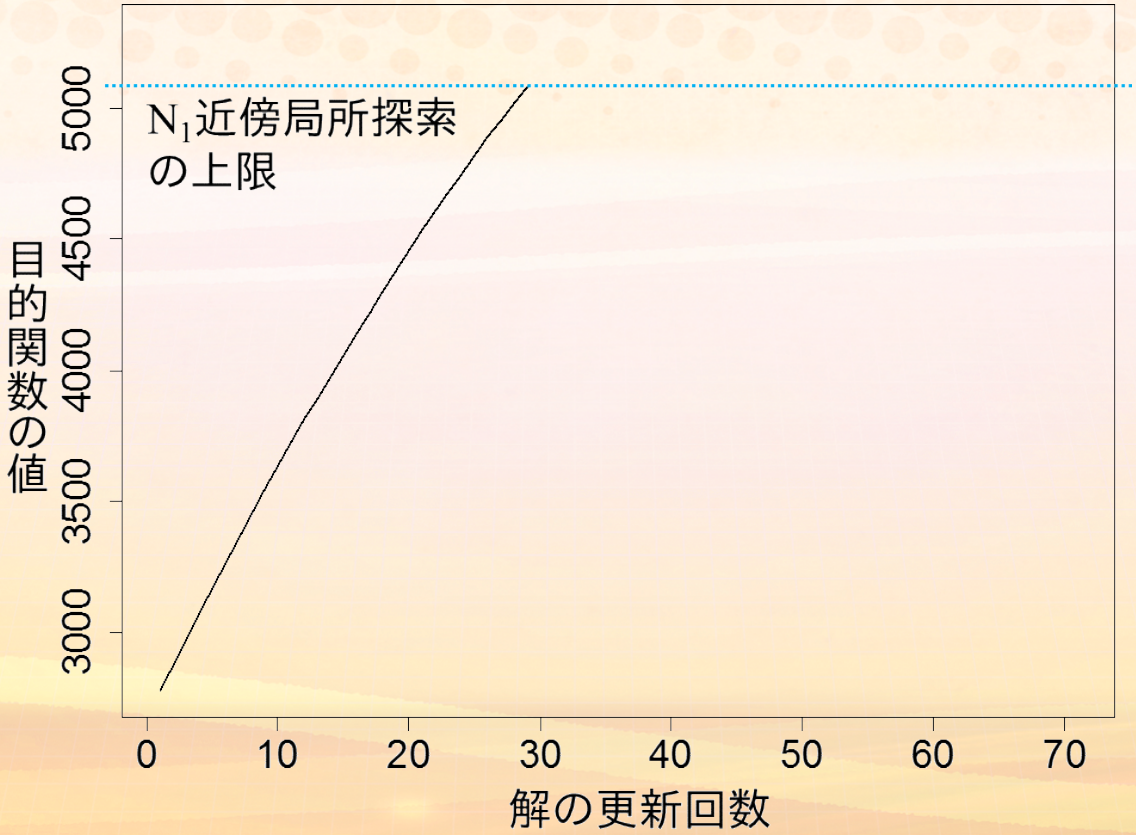
◆k-opt近傍

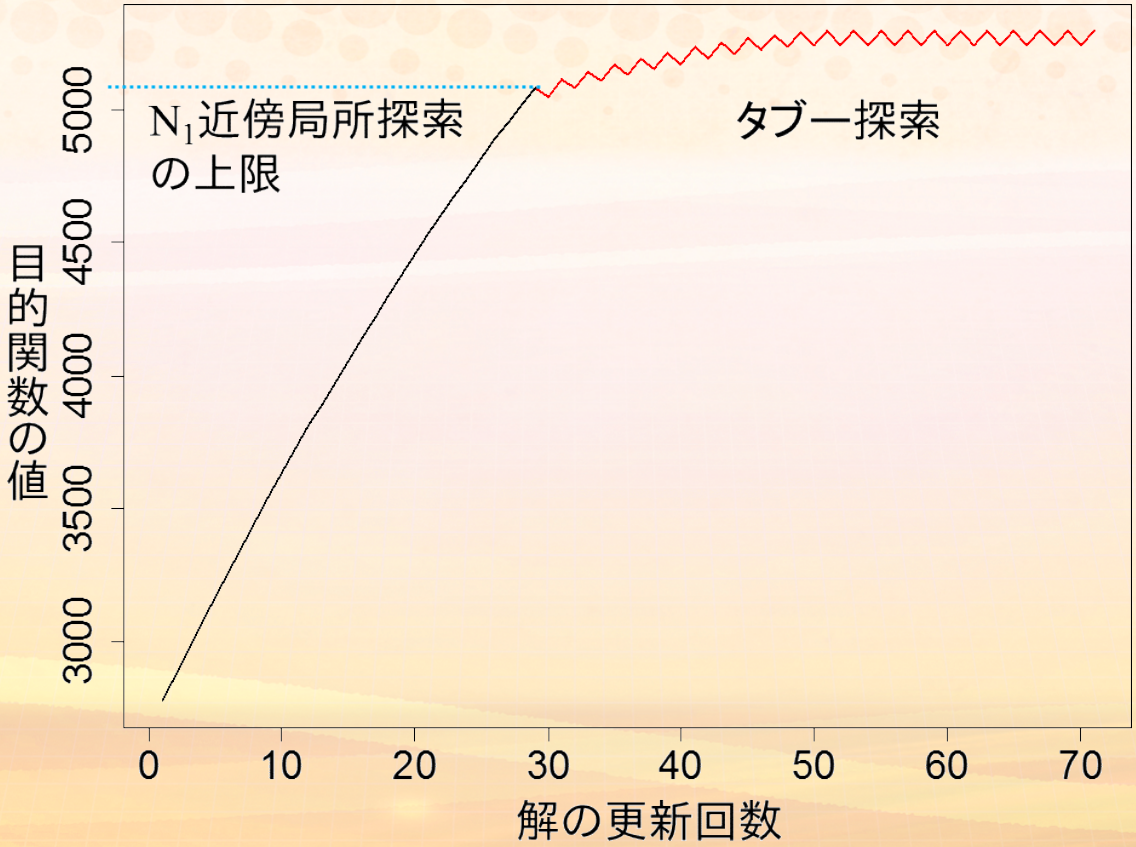
- 巡回路から隣り合わない3本の枝を取り除いて，別の3本の枝を加える3-opt近傍は，1つの巡回路に対して$O(n^3)$存在
- 巡回路から隣り合わないk本の枝を取り除いて，別のk本の枝を加えるk-opt近傍は，1つの巡回路に対して$O(n^k)$存在
- 計算量と解の精度から2-opt近傍や3-opt近傍が用いられることが多い

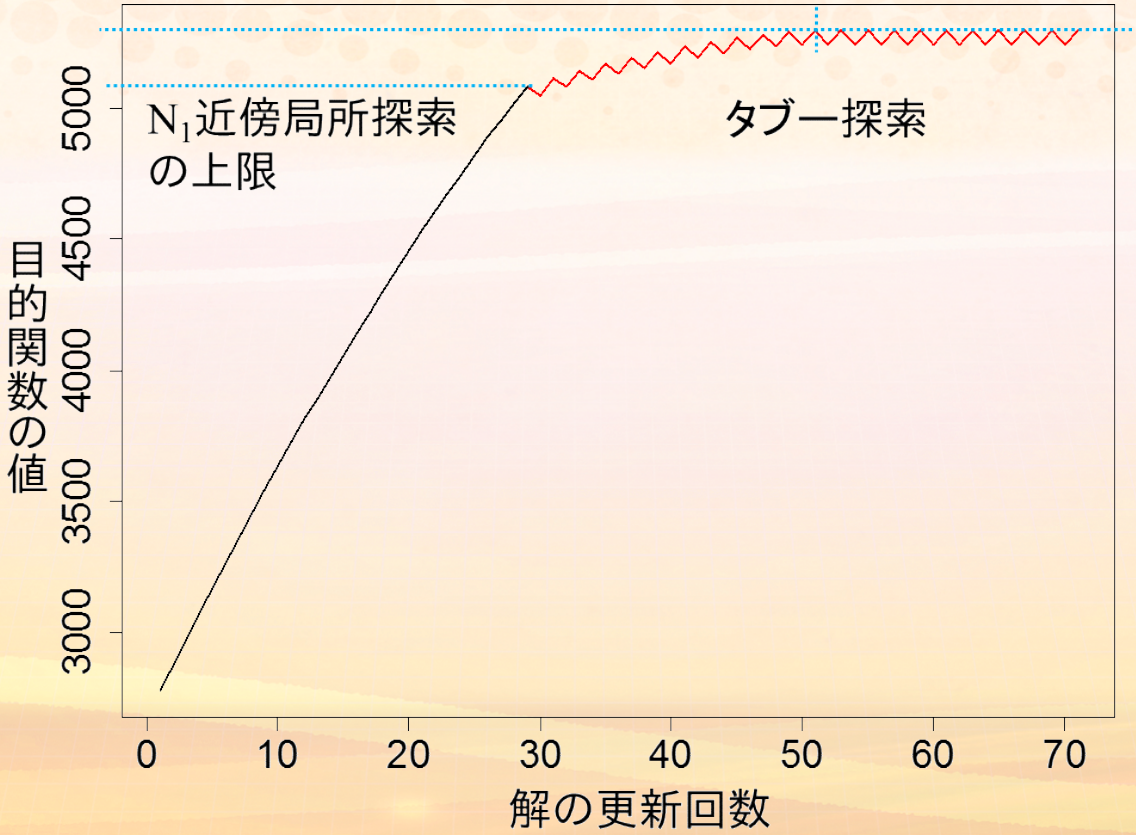
タブー探索法

タブー探索法

- $N(\boldsymbol{x})$に$\boldsymbol{x}$より良い解が見つからない時，改悪になっても解の更新を行う
- 探索の履歴をタブーリストに記録しておき，同じ解を繰り返し探索することを防ぐ
- タブーリストが巨大化すると計算量が大きくなるので，古い情報は削除してリストの巨大化を防ぐ


N_1近傍局所探索
の上限
目的関数の値
3000
3500
4000
4500
5000
0
10
20
30
40
50
60
70
解の更新回数


目的関数の値
3000
3500
4000
4500
5000
0
10
20
30
40
50
60
70
解の更新回数
N_1近傍局所探索
の上限
タブー探索


N_1近傍局所探索
の上限
タブー探索
目的関数の値
3000
3500
4000
4500
5000
0
10
20
30
40
50
60
70
解の更新回数

タブー探索法

(0) Lを空に，Lに保持する履歴の最大値をnにする
初期解$\boldsymbol{x}$を選ぶ

(1) $N(\boldsymbol{x})$において，$\boldsymbol{x}$自身とLの要素以外で
最も良い解$\boldsymbol{x}'$を見つける

(2) $\boldsymbol{x}$をLに加える
Lの要素数がnを超えたら，最も古い要素を取り除く
$\boldsymbol{x} \leftarrow \boldsymbol{x}'$とする

(3) 終了条件を満たせば，$\boldsymbol{x}$およびLの要素の中から
最も良い解を出力して終了
そうでなければ，(1)へ

タブー探索法

◆終了条件

- $f(\boldsymbol{x})$の値が一定以下に達する
- 反復回数が一定回数に達する
- より良い解に更新されないことが一定回数繰り返される

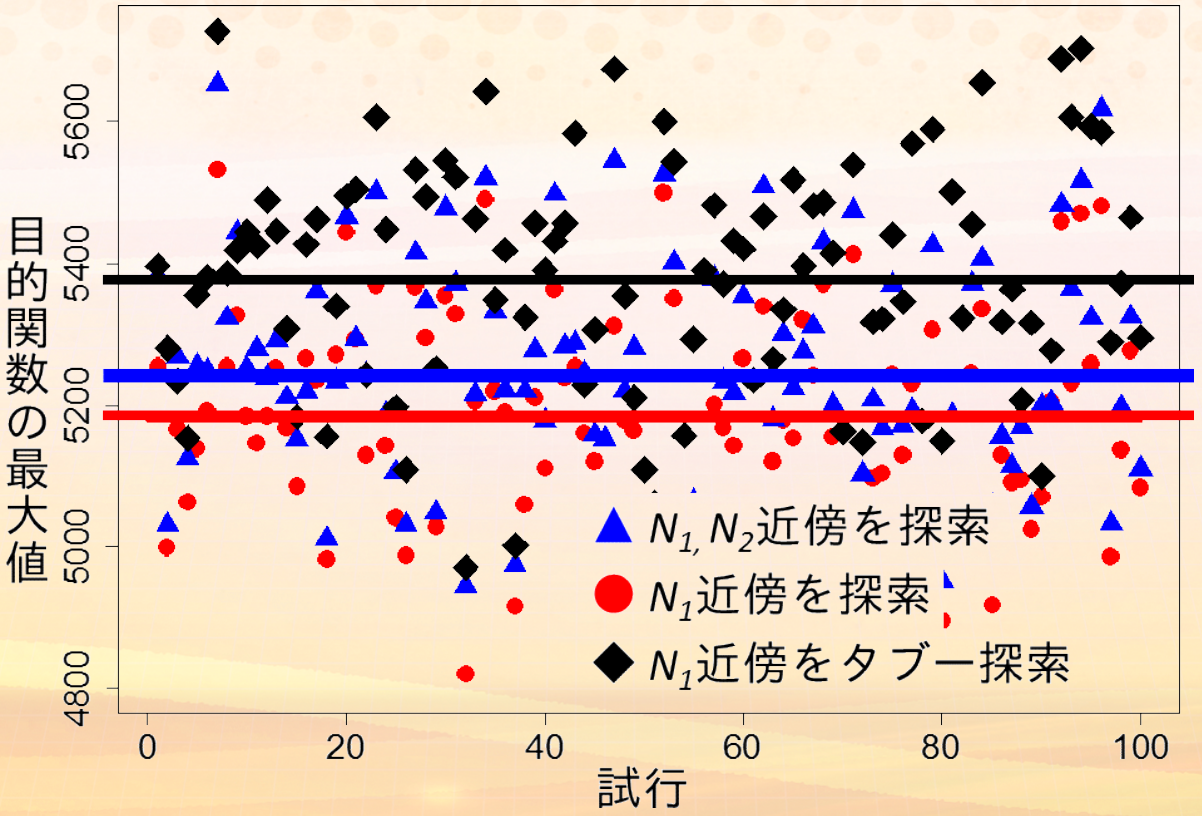
タブー探索法

◆ナップサック問題への適用

- 100変数ナップサック問題100問

1. 欲張り法 + N_1近傍探索
2. 欲張り法 + $N_1 \cup N_2$近傍探索
3. 欲張り法 + N_1近傍探索 + タブー探索法

- タブーリストのサイズ 7
- 20回連続で最良解を更新しないと終了


目的関数の最大値
4800
5000
5200
5400
5600
0
20
40
60
80
100
試行
N_1, N_2近傍を探索
N_1近傍を探索
N_1近傍をタブー探索

遺伝的アルゴリズム

遺伝的アルゴリズム

- 環境に適合した形質を持つ個体が生き残り，多くの子孫を残す … 淘汰（とうた）
- 子は両親の形質を一部ずつ受け継ぐ … 交叉（こうさ）
  より環境に適合した形質をもつ個体が出現する可能性
- 突然変異により新たな特性を持つ個体が出現する … 突然変異
- 解を個体に見立てて，淘汰，交叉，突然変異を繰り返し，解を進化させる

遺伝的アルゴリズム

- 解である個体は遺伝子の文字列として表現される

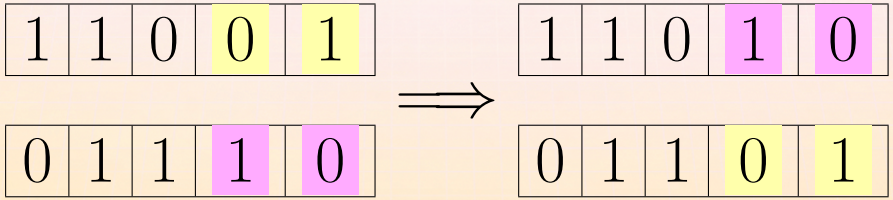
0	1	2	0	1

- 解の質は個体の適合度として表される
- 適合度は目的関数の値を反映
- 適合度の高い個体ほど生き残り子孫を残す確率が高く，適合度の低い個体は死滅しやすい … 淘汰
- 淘汰は個体の集団内での適合度の分布にしたがい確率的に決定

遺伝的アルゴリズム

◆交叉

- 2つの個体間で遺伝子の組換えによって新しい個体を生成

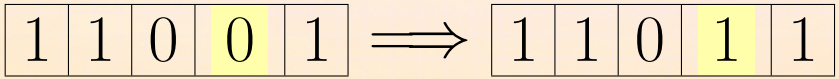


- 切断される箇所が1か所 … 1点交叉

遺伝的アルゴリズム

◆突然変異

- 1つの個体の遺伝子の一部の値を別の値に置き換えることにより新しい個体を生成
- 集団内での遺伝子の多様性を保つ効果



遺伝的アルゴリズム

(0) 個体を N 個生成
(1) 各個体の適合度を計算
(2) 適合度に応じた確率で個体を選択
遺伝的操作（交叉，突然変異）
次世代の個体を N 個生成
(3) 現世代の個体群を次世代の個体群で置き換える
(4) 終了基準を満たせば，最も適合度の高い解を
出力して，終了
そうでなければ，(1) へ

遺伝的アルゴリズム

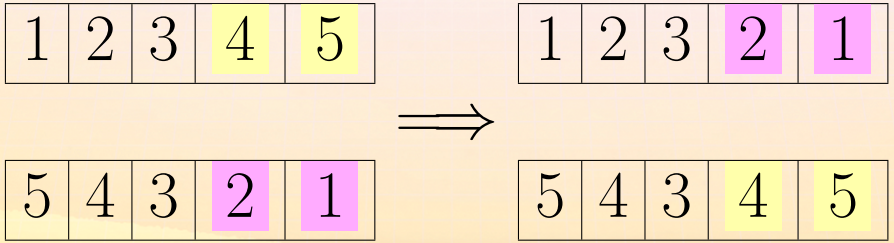
(2) 適合度に応じて定められた確率で個体を選択し，
それを基に次世代の個体を生成
定められた確率で以下のいずれかの遺伝的操作を
次世代の個体数がN個に達するまで繰り返す

- 2個体を選択して交叉
- 1個体を選択して突然変異
- 1個体を選択して逆位
- 1個体を選択してコピー

遺伝的アルゴリズム

◆TSPへの適用

A	B	C	D	E
1	4	2	5	3



遺伝的アルゴリズム

- $D \to B \to A \to E \to C$
- 都市 A, B, C, D, E を $1, 2, 3, 4, 5$ と順序づける
- 最初に訪れる D の順序 4 を遺伝子の文字列(遺伝子コード)の最初の位置(遺伝子座)に入れる
- 都市 D を取り除き，残る4都市 A, B, C, E を $1, 2, 3, 4$ と順序づける

都市	順序	遺伝子コード
D	ABCDE 12345	4
B	ABCE 1234	2
A	ACE 123	1
E	CE 12	2
C	C 1	1

遺伝的アルゴリズム

- $D \to B \to A \to E \to C$
- 残る4都市A, B, C, Eを$1, 2, 3, 4$と順序づける
- 2番目に訪れるBの順序2を2番目の遺伝子座に入れる
- 都市Bを取り除き，残る3都市A, C, Eを$1, 2, 3$と順序づける

都市	順序	遺伝子コード
D	ABCDE 12345	4
B	ABCE 1234	2
A	ACE 123	1
E	CE 12	2
C	C 1	1

遺伝的アルゴリズム

- $D \to B \to A \to E \to C$
- 残る3都市A, C, Eを$1,\ 2,\ 3$と順序づける
- 3番目に訪れるAの順序1を3番目の遺伝子座に入れる
- 都市Aを取り除き，残る2都市C, Eを$1, 2$と順序づける

都市	順序	遺伝子コード
D	ABCDE 12345	4
B	ABCE 1234	2
A	ACE 123	1
E	CE 12	2
C	C 1	1

遺伝的アルゴリズム

- $D \to B \to A \to E \to C$
- 残る2都市C, Eを$1, 2$と順序づける
- 4番目に訪れるEの順序2を4番目の遺伝子座に入れる
- 都市Eを取り除き，残る1都市Cを1と順序づける
- 5番目に訪れるCの順序1を5番目の遺伝子座に入れる

都市	順序	遺伝子コード
D	ABCDE 12345	4
B	ABCE 1234	2
A	ACE 123	1
E	CE 12	2
C	C 1	1

遺伝的アルゴリズム

$$D \to B \to A \to E \to C$$

4	2	1	2	1

都市	順序	遺伝子コード
D	ABCDE 12345	4
B	ABCE 1234	2
A	ACE 123	1
E	CE 12	2
C	C 1	1

遺伝的アルゴリズム

◆適合度関数

- 適合度を計算する適合度関数は目的関数を反映
- TSPの目的関数は巡回路の総移動距離 … 最小化
- 個体iの遺伝子表現を$\boldsymbol{x}_i$，目的関数を$f(\boldsymbol{x}_i)$
  個体iの適合度f_iは，例えば，

$$f_i = 1/f(\boldsymbol{x}_i)$$

遺伝的アルゴリズム

- 適合度の高い個体を高い確率で次世代に残す
- 個体iを選択する確率p_iは，例えば，

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

- 確率分布に応じて個体数Nの回数だけ乱数を発生
- 発生した数iに相当する個体iを重複を許して
  次の世代の個体として選択

遺伝的アルゴリズム

- 個体iを選択する確率p_i

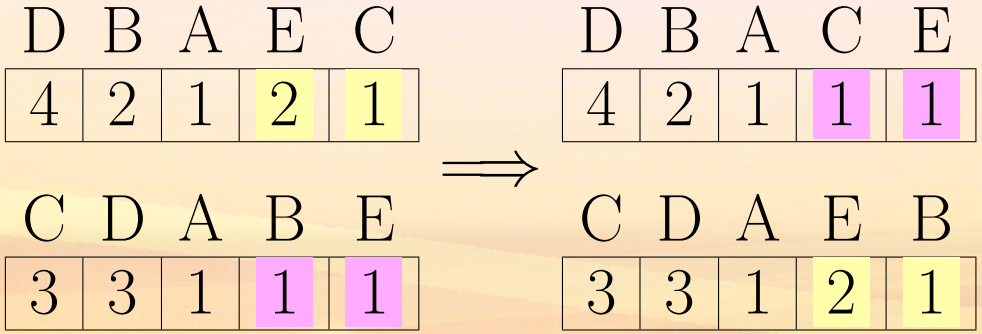
$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

- 確率分布に応じて個体数Nの回数だけ乱数を発生
- 発生した数iに相当する個体iを重複を許して次の世代の個体として選択（ルーレット方式）

遺伝的アルゴリズム

◆交叉

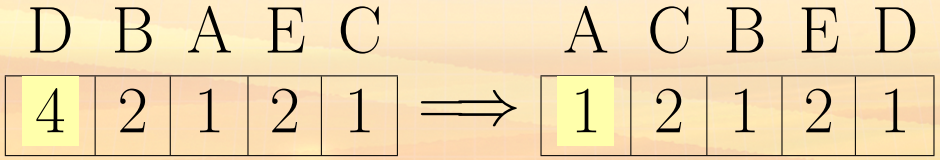
- 個体の集団から選んだ2個体に対して，交叉を行う遺伝子座の位置をランダムに決め，
  その位置以降の遺伝子を入れ替える



遺伝的アルゴリズム

◆突然変異

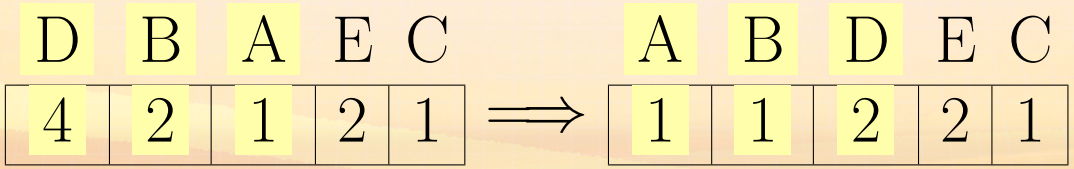
- 個体の集団から選んだ1個体に対して，
  突然変異を行う遺伝子座の位置をランダムに決め，
  その位置の遺伝子の値を（とり得る値に）
  ランダムに変更する
- 突然変異が起こる確率は小さく設定する



遺伝的アルゴリズム

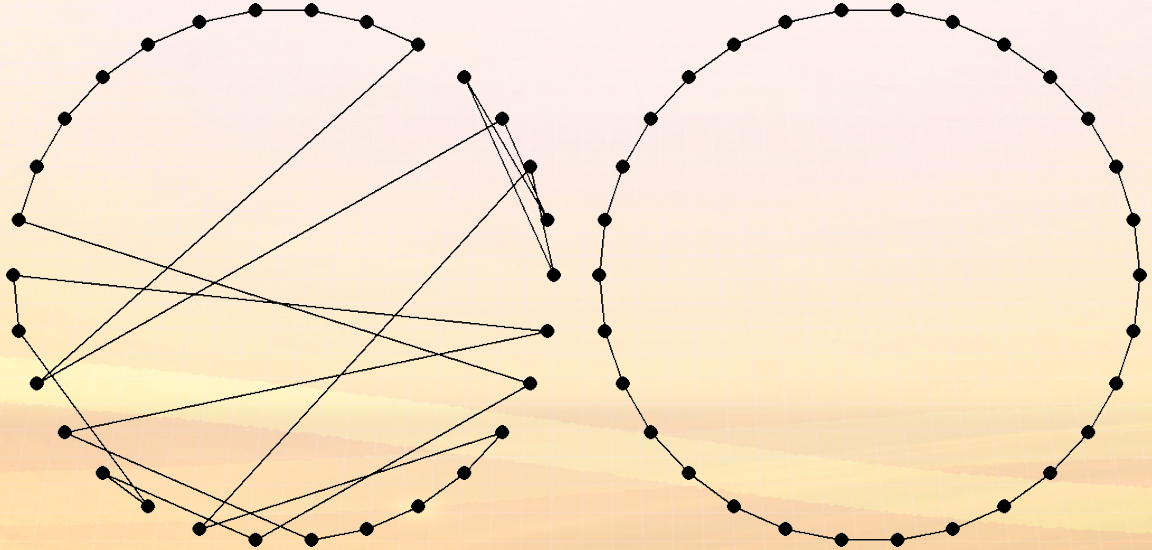
◆逆位

- 個体の集団から選んだ1個体に対して，逆位の開始位置と終了位置をランダムに決め，その間の都市の訪問順が逆になるように遺伝子の値を変更する
- 逆位は突然変異の一種と分類されることもある



遺伝的アルゴリズム

- 円周上に30都市を等間隔に配置したTSP
- 最適解は正30角形の巡回路（ほぼ円）

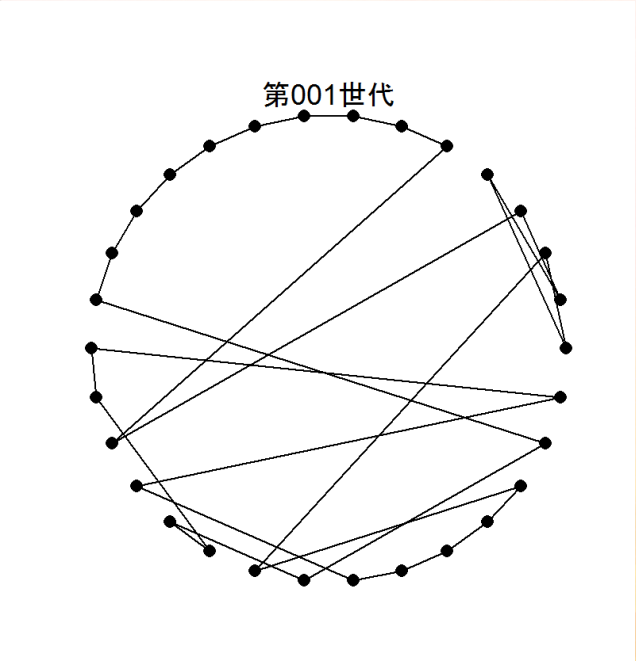


遺伝的アルゴリズム

- 円周上に30都市を等間隔に配置したTSP
- 個体数 400
- 適合度上位4個体を次世代にコピー
- 8個体を突然変位
- 8個体を逆位
- 380個体を交叉 (一点交叉)
- ルーレット方式

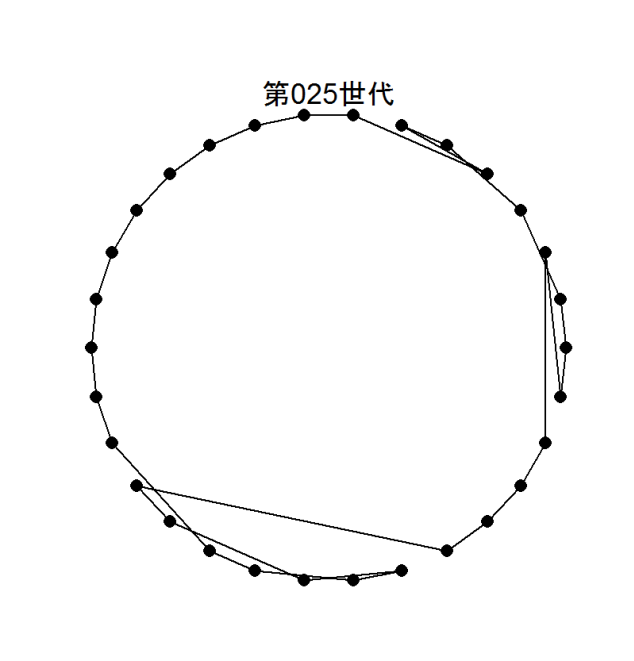
遺伝的アルゴリズム

第1世代(初期解)　距離：21.71



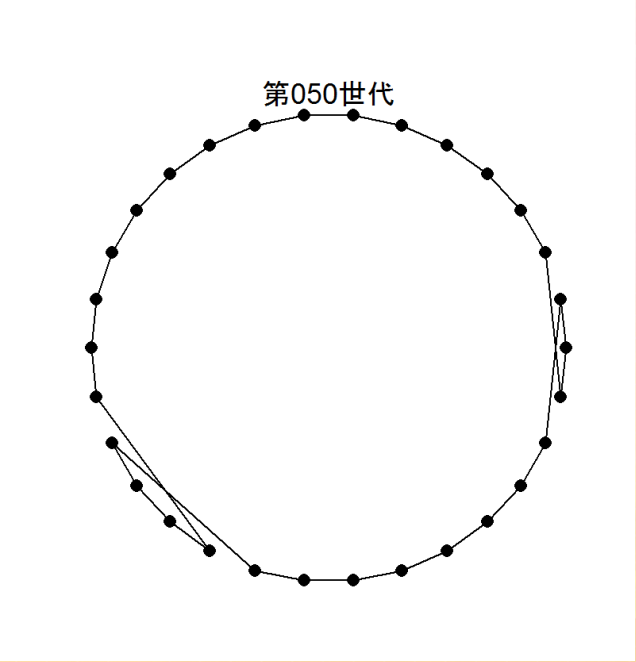
遺伝的アルゴリズム

第25世代　距離：10.68



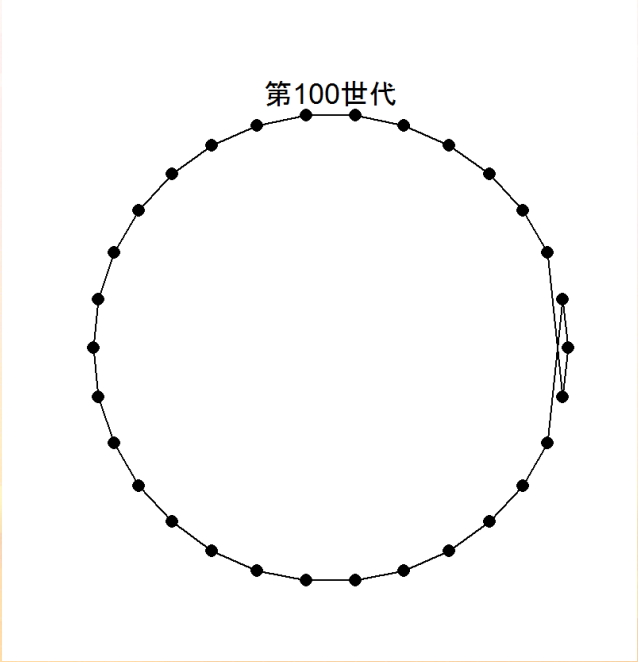
遺伝的アルゴリズム

第50世代　距離：8.30



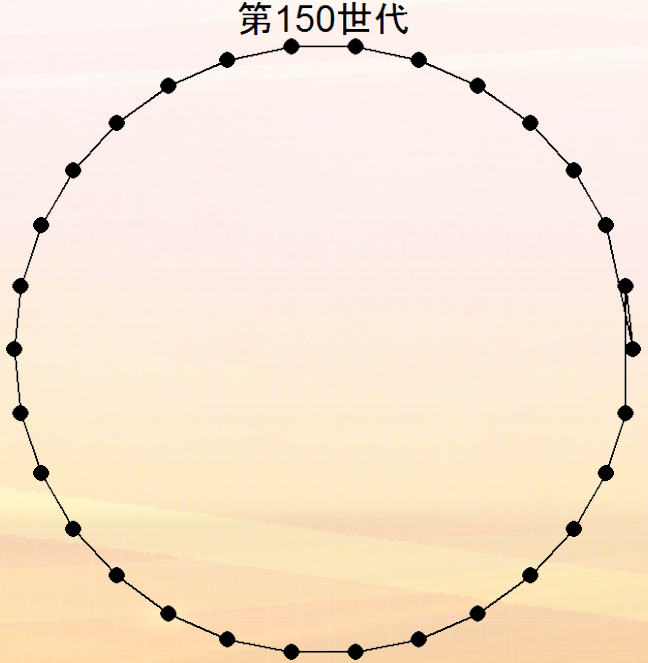
遺伝的アルゴリズム

第100世代　距離：7.08



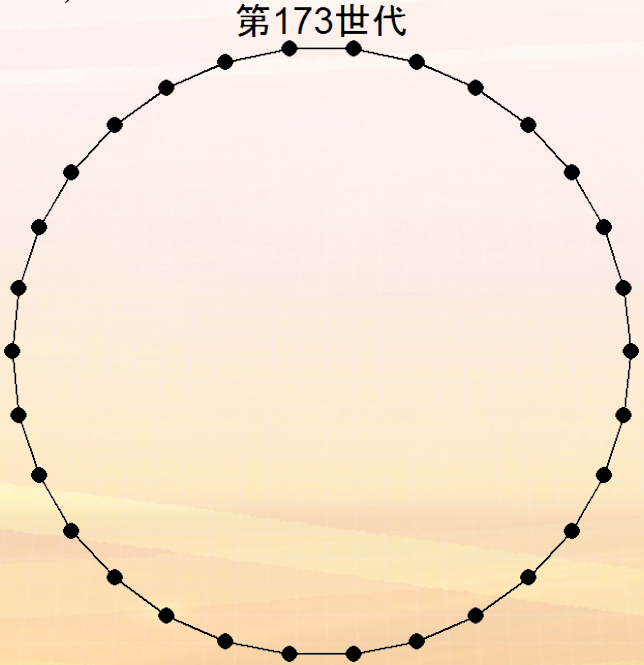
遺伝的アルゴリズム

第150世代　距離：6.69



遺伝的アルゴリズム

第173世代(最適解)　距離：6.27



遺伝的アルゴリズム