



# 遺伝的アルゴリズム (Genetic Algorithm) を始めよう!

2014年11月17日

先端IT活用推進コンソーシアム  
クラウド・テクノロジー活用部会  
勉強会資料

岡村 和英 (株式会社テクリエ)

# Introduction

# Genetic Algorithms Programming Library for JavaScript



<https://github.com/kzokm/ga.js>

## Samples of Genetic Algorithm



<http://ga-samples.herokuapp.com/>

# Today's Topics

What's GA  
Merits of GA  
Genetic Glossary  
Genetic Algorithms  
Chromosome Encodings  
Encoding Examples  
Genetic Operations  
Selection, Crossover, Mutation  
New Generation Alternation Model  
Known Issues  
Summary

# What's GA



生物進化における遺伝と  
適者生存による自然淘汰の仕組みを  
ソフトウェア的に模すことで  
複雑な問題に対する最適解を  
探索する手法



ある命題に対する解の候補を  
遺伝子(gene)とその集合体である  
染色体(chromosome)で表現した  
個体(individual)を複数用意し、  
適応度(fitness)の高い個体を優先して  
交叉(crossover)、突然変異(mutation)  
などの操作を繰り返しながら  
最適解の探索をおこなう



# N700系新幹線



フロントノーズ（エアロ／ダブル  
ウイング）の空力設計に際し、  
GAを用いて約5000パターンのコン  
ピュータシミュレーションを行った  
結果からデザインを決定した。



# ST5計画



NASAのニュー・ミレニアム計画の一環として行われた技術試験計画 Space Technology 5 に用いられた3機の人工衛星に搭載されたアンテナの形状は、GAを用いて設計された。



# Merits of GA



評価関数の可微分性や单峰性などの知識がない場合であっても適用可能。



必要とされる条件は評価関数の  
**全順序性**と、探索空間が**位相**  
(topology)を持っていること。



遺伝子の表現の仕方によって、  
**組合せ最適化問題**や**NP困難**な  
問題などのさまざまな問題に対  
する適用が可能。



# Genetic Glossary

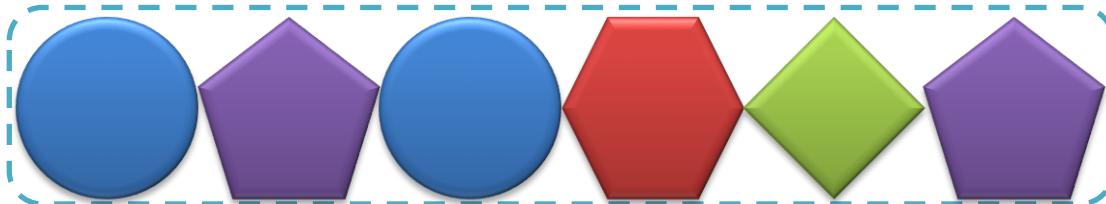




# 遺伝子(gene)

個体の形質を表すための  
基本となる構成要素

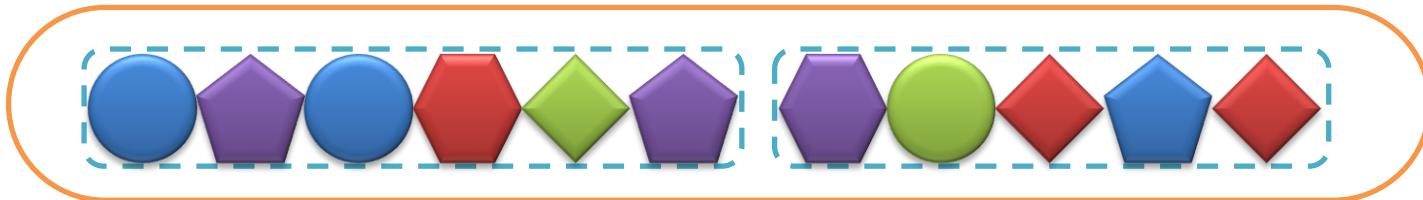




染色体(chromosome)

複数の遺伝子の集まり

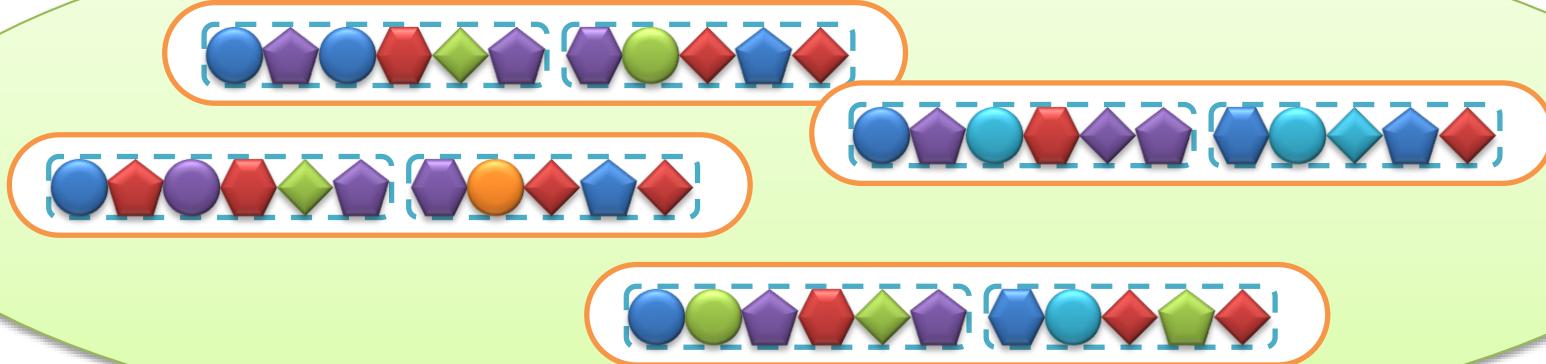




## 個体(individual)

1つまたは複数の染色体によって表現される自律的な個命題に対する解の候補

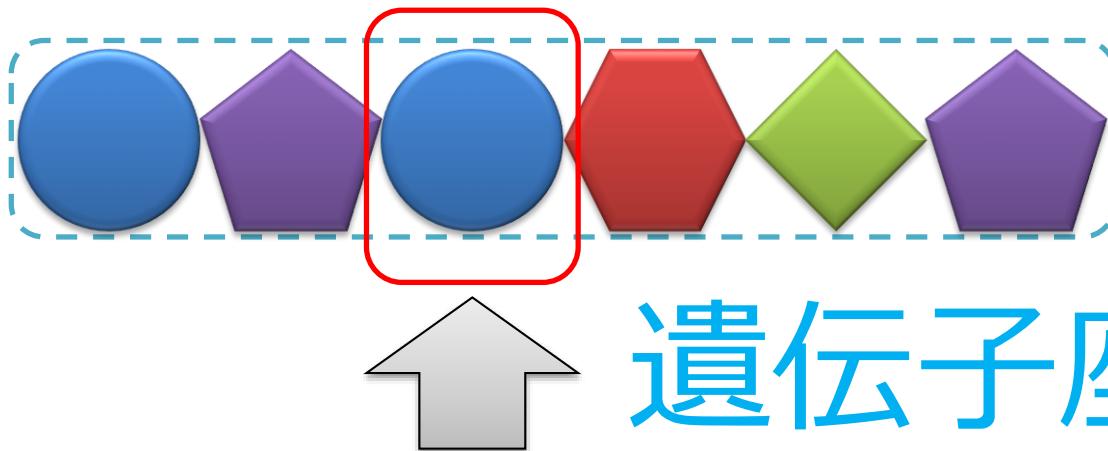




# 集団 (population)

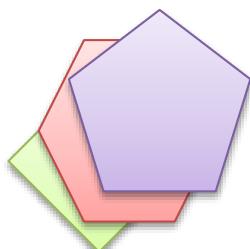
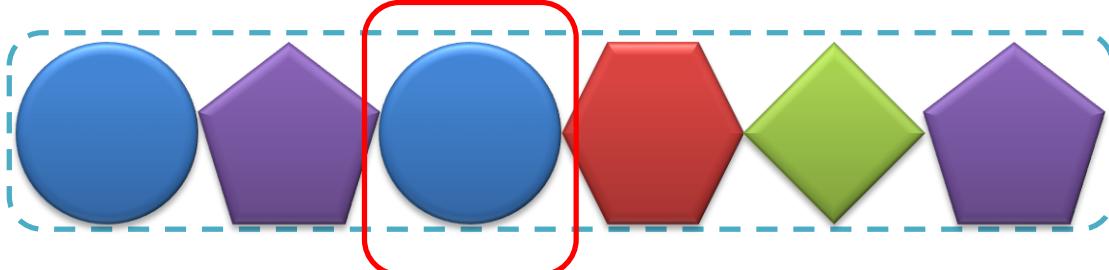
様々な個体の集まり





染色体上における各遺伝子の  
位置

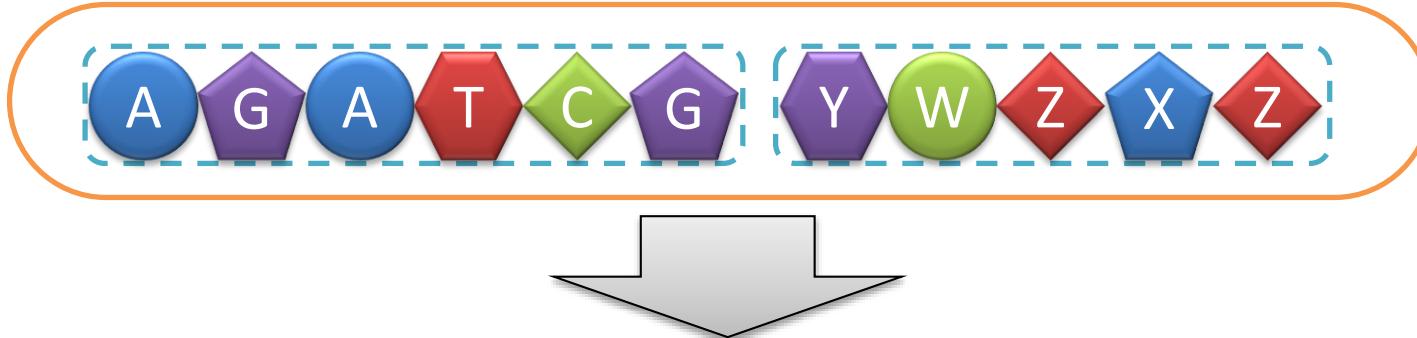




## 対立遺伝子(allele)

ある遺伝子座において遺伝子  
がとりうる別の値





遺伝子型(因子型; genotype)

遺伝子を用いた内部表現



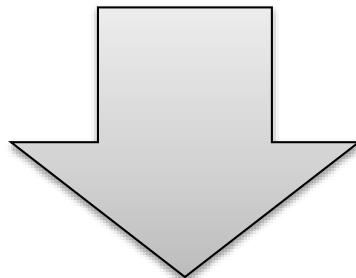


## 表現型(phenotype)

遺伝子によって発現する  
形質の外部表現



くちばし=赤・三角、羽=茶色



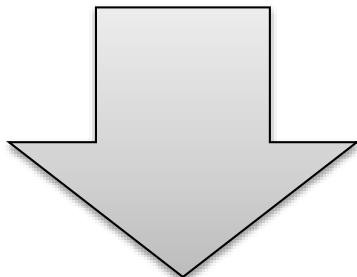
符号化(encoding)

AGATCG-YWZXZ

表現型から遺伝子型への変換



AGATCG-YWZXZ



復号化(decoding)

くちばし=赤・三角、羽=茶色

遺伝子型から表現型への変換



# Genetic Algorithms



1. 解を表現する**符号化方法**を決定する。



1. 解を表現する符号化方法を決定する。
2. **N個のランダムな個体**を含む集団を作成する。これを「現世代」と呼ぶ。



1. 解を表現する符号化方法を決定する。
2.  $N$ 個のランダムな個体を含む集団を作成する。これを「現世代」と呼ぶ。
3.  $N$ 個の個体を格納可能な集団を用意する。  
これを「次世代」と呼ぶ。



1. 解を表現する符号化方法を決定する。
2.  $N$ 個のランダムな個体を含む集団を作成する。これを「現世代」と呼ぶ。
3.  $N$ 個の個体を格納可能な集団を用意する。これを「次世代」と呼ぶ。
4. **評価関数を用いて、現世代の各個体の適応度をそれぞれ計算する。**



2. N個のランダムな個体を含む集団を作成する。これを「現世代」と呼ぶ。
3. N個の個体を格納可能な集団を用意する。これを「次世代」と呼ぶ。
4. 評価関数を用いて、現世代の各個体の適応度をそれぞれ計算する。

5. ある確率で次の3つの操作を行い、その結果を次世代に保存する。

- 1) 現世代から2つの個体を選択する。
- 2) 選択された個体を用いて交叉を行い、子孫(offspring)を生成する。
- 3) 交叉によって生成された子孫に対して、突然変異を適用する。



2.  $N$ 個のランダムな個体を含む集団を作成する。これを「現世代」と呼ぶ。
  3.  $N$ 個の個体を格納可能な集団を用意する。これを「次世代」と呼ぶ。
  4. 評価関数を用いて、現世代の各個体の適応度をそれぞれ計算する。
  5. ある確率で次の3つの操作を行い、その結果を次世代に保存する。
    - 1) 現世代から2つの個体を選択する。
    - 2) 選択された個体を用いて交叉を行い、子孫(offspring)を生成する。
    - 3) 交叉によって生成された子孫に対して、突然変異を適用する。
- 6. 次世代の個体数が $N$ 個になるまで5の操作を繰り返す。**



- 評価関数を用いて、現世代の各個体の適応度を計算します。
5. ある確率で次の3つの操作を行い、その結果を次世代に保存する。
    - 1) 現世代から2つの個体を選択する。
    - 2) 選択された個体を用いて交叉を行い、子孫(offspring)を生成する。
    - 3) 交叉によって生成された子孫に対して、突然変異を適用する。
  6. 次世代の個体数がN個になるまで5の操作を繰り返す。
7. 次世代を現世代に移行し、**命題が収束するまで3~6の手順を繰り返す**。この結果を「**最終世代**」と呼ぶ。  
命題の収束を判定する方法として・・・



7. 次世代を現世代に移行し、命題が収束するまで3~6の手順を繰り返す。この結果を「最終世代」と呼ぶ。

命題の収束を判定する方法として、以下のような場合が用いられる。

- ・ 集団中の最大適応度が、ある閾値より大きくなった場合
- ・ 集団全体の平均適応度が、ある閾値より大きくなった場合
- ・ 集団の適応度増加率が、ある閾値以下の世代が一定期間続いた場合
- ・ 世代交代の回数が規定の最大世代数に達した場合



次世代を現世代に移行し、命題が収束するまで、繰り返す。

この結果を「最終世代」と呼ぶ。

命題の収束を判定する方法として、以下のような場合が用いられる。

- ・集団中の最大適応度が、ある閾値より大きくなった場合
- ・集団全体の平均適応度が、ある閾値より大きくなった場合
- ・集団の適応度増加率が、ある閾値以下の世代が一定期間続いた場合
- ・世代交代の回数が規定の最大世代数に達した場合

## 8. 最終世代の中で最も適応度の高い個体 を「解」として出力する。



# Chromosome Encodings



解の特徴を  
遺伝子として表現する



バイナリエンコーディング  
(binary encoding)

順列エンコーディング  
(permutation encoding)

実数値エンコーディング  
(real encoding)

木構造エンコーディング  
(tree encoding)



# バイナリエンコーディング (binary encoding)

順列エンコーディング  
(permutation encoding)

実数値エンコーディング  
(real encoding)

木構造エンコーディング  
(tree encoding)



# バイナリエンコーディング (binary encoding)

各遺伝子を0, 1のビットとして表現する符号化方法



# バイナリエンコーディング (binary encoding)

染色体A 1001011101101  
染色体B 1010011010111



バイナリエンコーディング  
(binary encoding)

順列エンコーディング  
(permutation encoding)

実数値エンコーディング  
(real encoding)

木構造エンコーディング  
(tree encoding)



# 順列エンコーディング (permutation encoding)

各遺伝子を順序を示す数字として表現する符号化方法



# 順列エンコーディング (permutation encoding)

染色体A	2	7	6	4	1	8	3	5	9
染色体B	4	6	1	9	5	7	3	8	2



# 順列エンコーディング (permutation encoding)

巡回セールスマン問題や仕事の順序などの並べ替え問題の表現に用いられる



バイナリエンコーディング  
(binary encoding)

順列エンコーディング  
(permutation encoding)

実数値エンコーディング  
(real encoding)

木構造エンコーディング  
(tree encoding)



# 実数値エンコーディング (real encoding)

各遺伝子を数値（または文字）として表現する符号化方法



# 実数値エンコーディング (real encoding)

染色体A 0.51 2.83 3.12 1.50

染色体B AGTCATGCAGCATTAA

染色体C 前進 前進 右 後退 左



# 実数値エンコーディング (real encoding)

実数値データなど、バイナリ  
エンコーディングでは表現が  
難しい場合に用いる



バイナリエンコーディング  
(binary encoding)

順列エンコーディング  
(permutation encoding)

実数値エンコーディング  
(real encoding)

木構造エンコーディング  
(tree encoding)



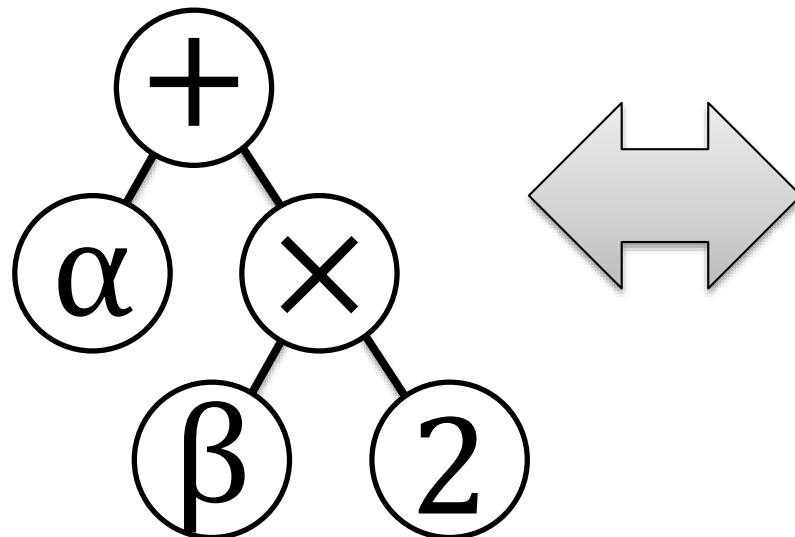
# 木構造エンコーディング (tree encoding)

染色体を 1 本の紐（配列）ではなく、木構造データとして表現する符号化方法



# 木構造エンコーディング (tree encoding)

染色体A



表現型

$$\alpha + (\beta \times 2)$$



# 木構造エンコーディング (tree encoding)

遺伝的プログラミング(Genetic Programming; GP)や進化的プログラミング(Evolutionary Programming; EP)などの表現に用いられる



# Encoding Examples



# ナップザック問題 (Knapsack Problem)

容量Cのナップザックが1つと、各々の価値がPi、容積がCiであるところのn個の品物Eiが与えられたとき、Cを超えない範囲でいくつかの品物をナップザックに詰め、入れた品物の価値の和を最大化するにはどの品物を選べばよいか？



ナップザック問題を  
バイナリエンコーディングで表現してみる



$n$ 個の荷物 → 遺伝子長( $L$ ) =  $n$

$i$ 番目の荷物 → 遺伝子座( $p$ ) =  $i$

荷物 $E_i$ を入れる → 遺伝子[ $i$ ] = 1

荷物 $E_i$ を入れない → 遺伝子[ $i$ ] = 0

適応度 → 持っている荷物の総価値



# ナップザック問題

荷物	E1	E2	E3	E4	E5	E6	E7	...	En
重さ( $C_i$ ) Kg	0.9	1.1	0.7	1.4	0.5	1.3	1.1		1.6
価値( $P_i$ ) \$	1.0	1.3	0.9	1.5	0.5	1.1	1.2		1.4
遺伝子表現	[ 1 0 0 1 0 1 0 ... 0 ]								
重さ $\leq C$ Kg	0.9		+	1.4	+	1.3	+	...	
適応度	1.0		+	1.5	+	1.1	+	...	



# 循環セールスマン問題 (Travelling Salesman Problem)

n個の都市 $C_i$ と、それぞれの都市間の距離 $D_{i,j}$ が与えられているとき、最初の都市を出発し、全ての都市を経由して、同じ都市に戻ってくるルートのうち、最短のルートを求めよ。



循環セールスマン問題を  
順列エンコーディングで表現してみる



$n$ 個の都市 → 遺伝子長( $L$ ) =  $n$

$i$ 番目に訪れる → 遺伝子座( $p$ ) =  $i$

都市 $C_x$  → 遺伝子[ $i$ ] =  $x$

適応度 → 循環ルートの総距離

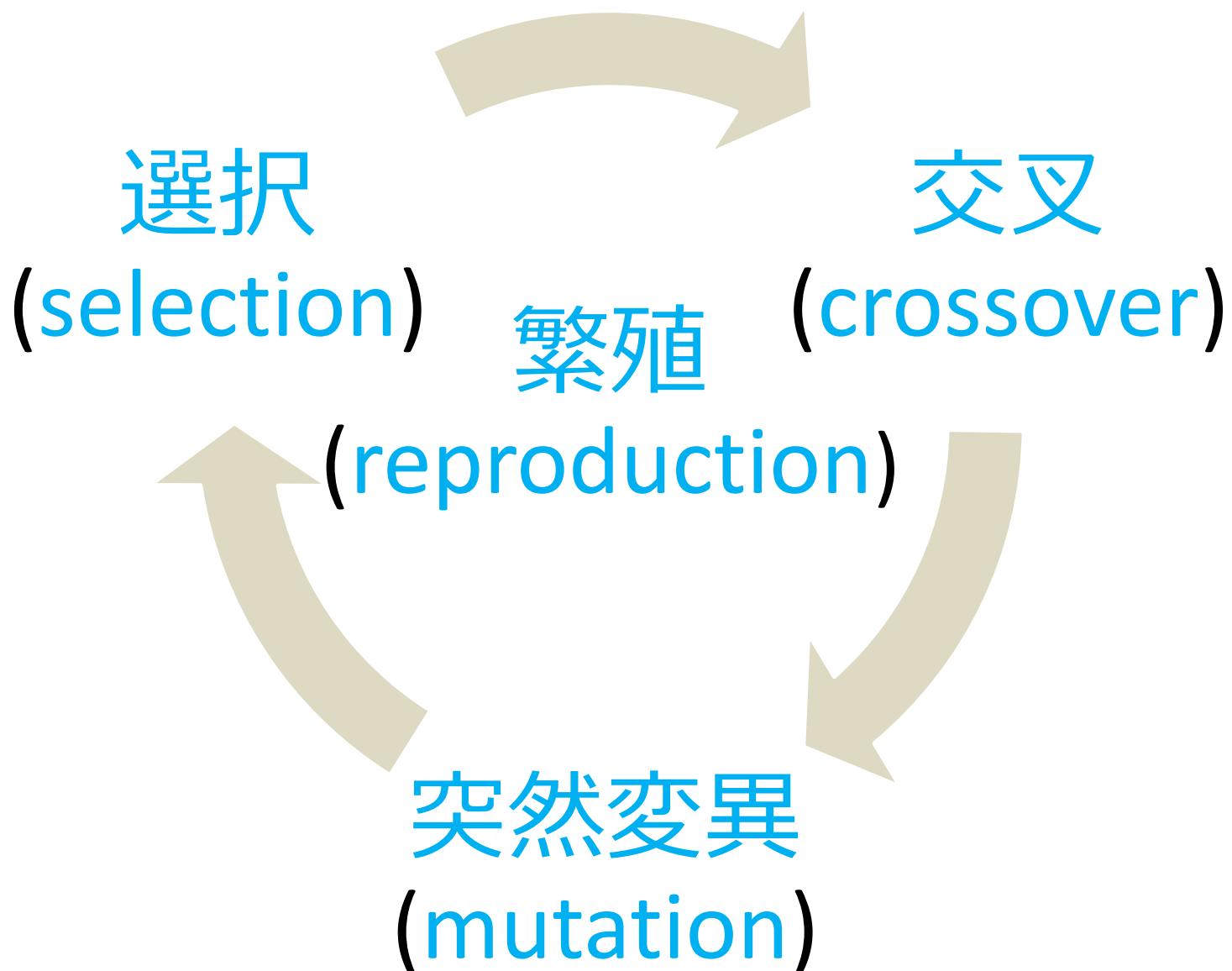


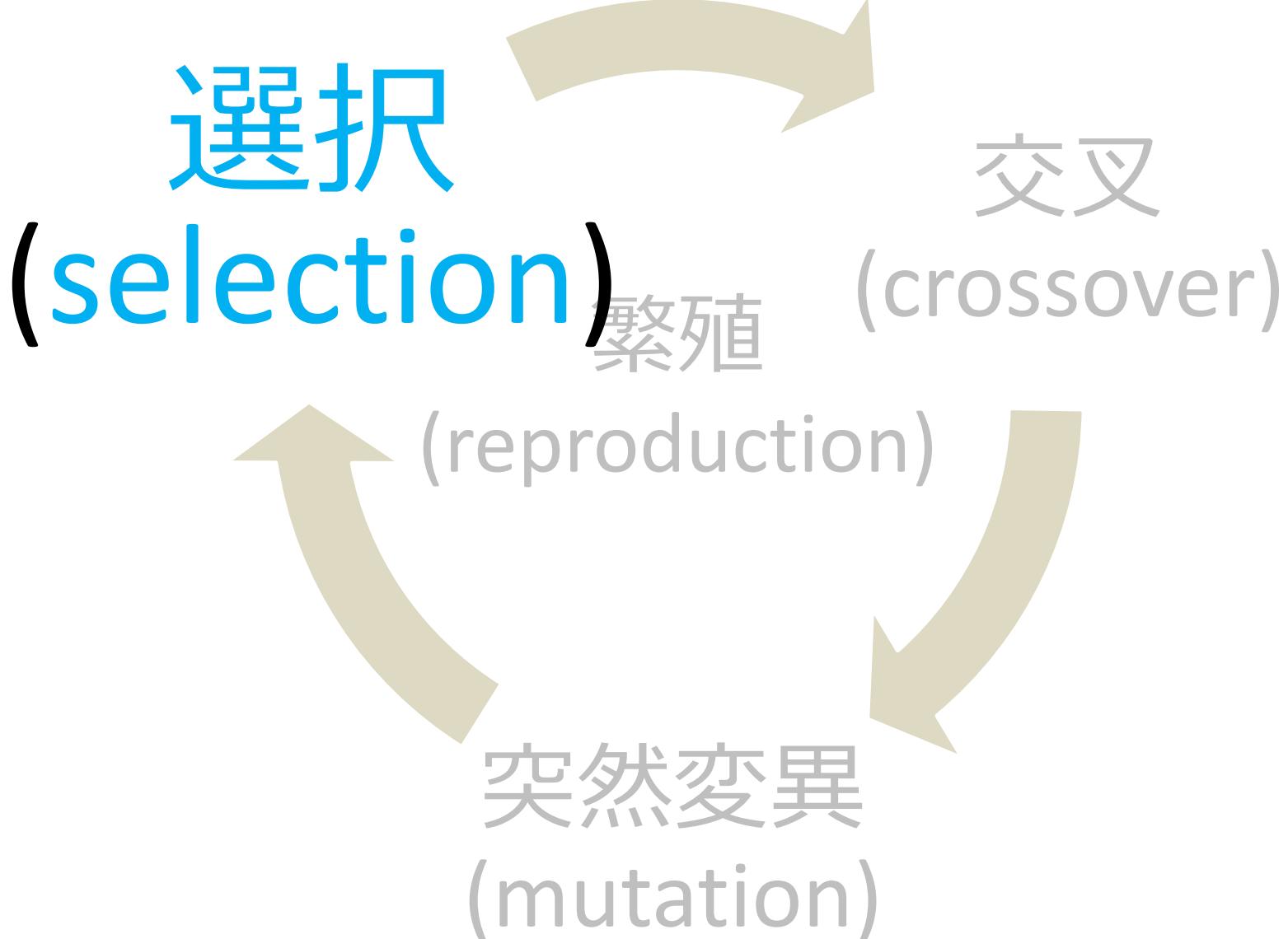
# 循環セールスマン問題



# Genetic Operations







# 選択 (selection)

適応度に基づいて、個体を増やしたり減らしたりするための対象を選び出す操作。

生物の自然淘汰をモデル化したもの。



ルーレット方式  
(roulette wheel selection)

ランキング方式  
(ranking selection)

トーナメント方式  
(tournament selection)

エリート主義  
(elitism)



ルーレット方式  
(roulette wheel selection)

ランキング方式  
(ranking selection)

トーナメント方式  
(tournament selection)

エリート主義  
(elitism)



# ルーレット方式 (roulette wheel selection)

集団をルーレット盤に見立て、ランダムな選択を行う方式。

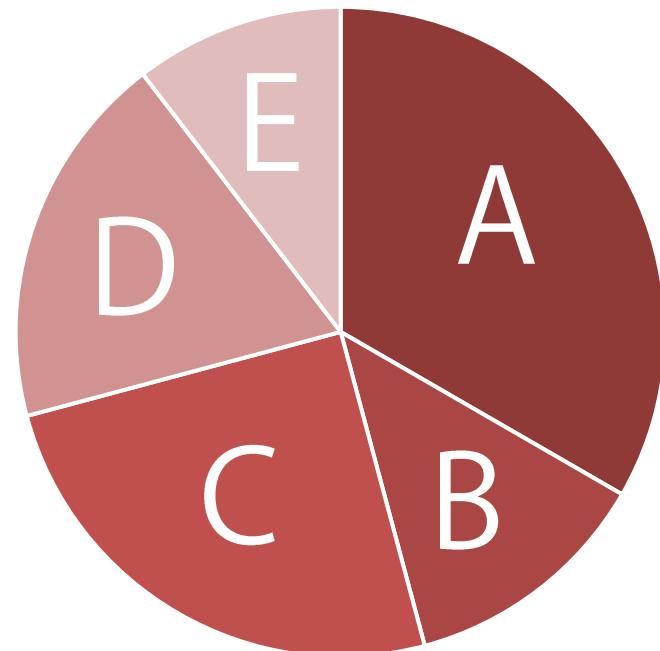
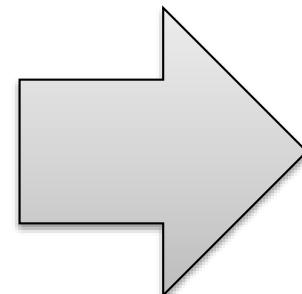
各個体の**適応度**が、ルーレット盤の面積と**比例**しており、適応度が高い染色体ほど選ばれる確率が高くなる。



# ルーレット方式 (roulette wheel selection)

個体 適応度

A	32
B	12
C	24
D	18
E	10



# ルーレット方式 (roulette wheel selection)

$$p_i = \frac{f(i)}{\sum_{n=1}^N f(n)}$$

$p_i$  := ある個体が選択される確率

$f(i)$  := 個体の適応度、 $N$  := 集団サイズ



# ルーレット方式 (roulette wheel selection)

適応度が正であることが前提。  
個体間の適応度の差が大きい場合、  
適応度の高い個体の選ばれる確率  
が高くなりすぎ、局所的な最適解  
への初期収束の原因となる。



ルーレット方式  
(roulette wheel selection)

ランキング方式  
(ranking selection)

トーナメント方式  
(tournament selection)

エリート主義  
(elitism)



# ランキング方式 (ranking selection)

ルーレット方式の変形。

適応度によって各個体のランク付けを行い、「1位なら確率 $p_1$ 、2位なら確率 $p_2$ 、……」と事前に決められた確率を適用する方式。



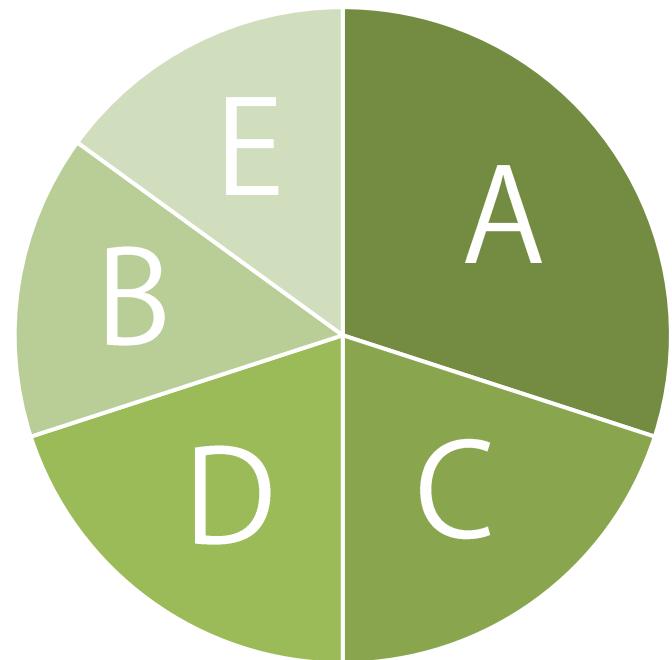
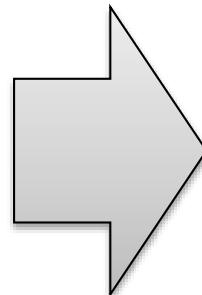
# ランキング方式 (ranking selection)

順位 選択確率

順位	選択確率
1	0.30
2	0.20
3	0.20
4	0.15
5	0.15

個体 適応度

A	32
B	12
C	24
D	18
E	10



# ランキング方式 (ranking selection)

個体間の適応度の差が選択確率に影響されないが、適応度に差がない個体であっても選択確率に大きな差が生じる可能性がある。

ランク付けを行うためには世代毎にソートを行うことが必要。



ルーレット方式  
(roulette wheel selection)

ランキング方式  
(ranking selection)

トーナメント方式  
(tournament selection)

エリート主義  
(elitism)



# トーナメント方式 (tournament selection)

予め決めた個体数（トーナメントサイズ）をランダムに抽出し、その中で最も適応度の高い個体を選択する方式。



# トーナメント方式 (tournament selection)

トーナメントサイズを変更することで、選択圧をコントロールすることが可能。

トーナメントサイズを大きくすることで選択圧を高めることができが、初期収束の原因となる。



ルーレット方式  
(roulette wheel selection)

ランキング方式  
(ranking selection)

トーナメント方式  
(tournament selection)

エリート主義  
(elitism)



# エリート主義 (elitism)

予め決めた個数の、最も適応度の高い個体をそのまま次世代にコピーする。

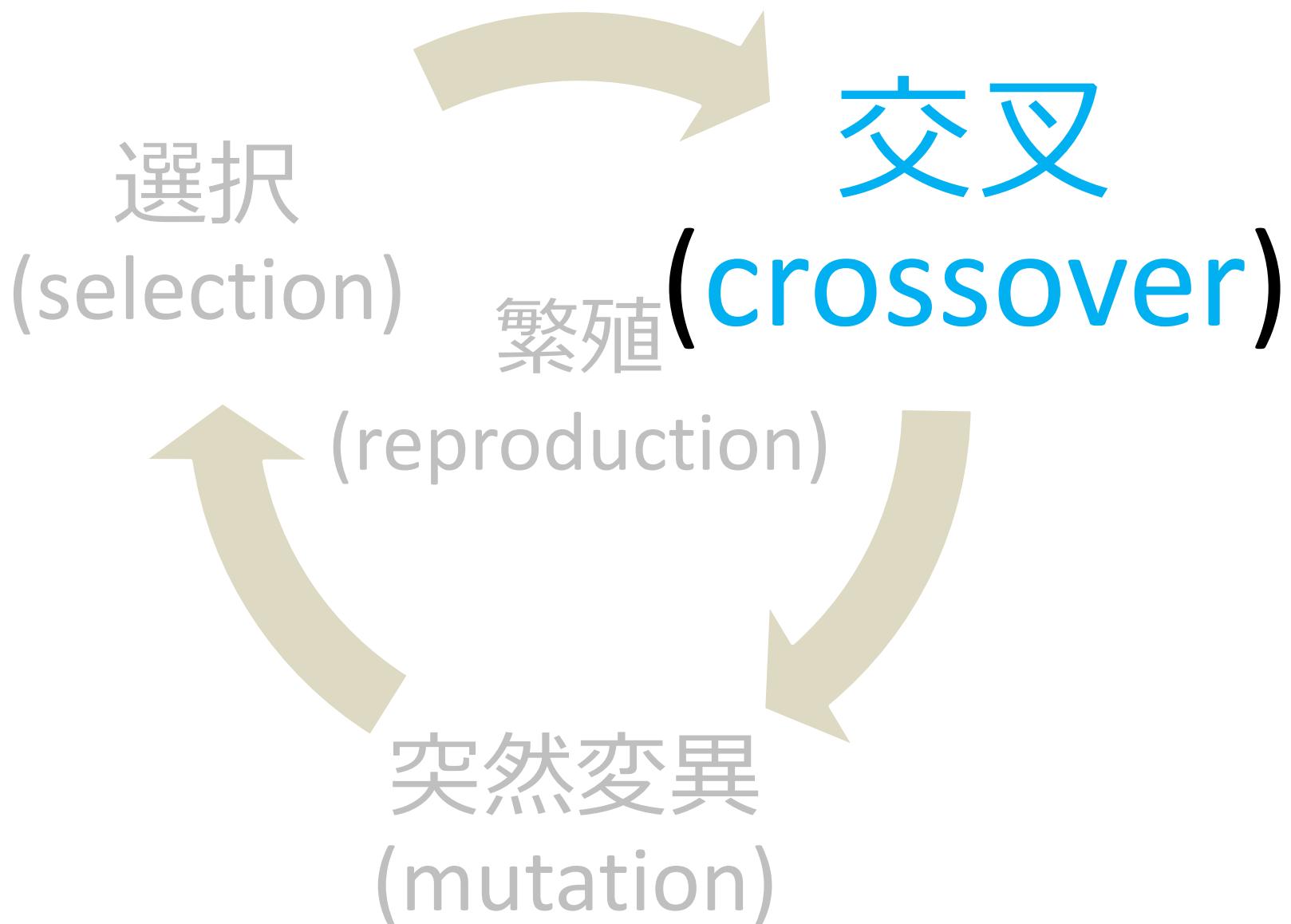


# エリート主義 (elitism)

世代間で適応度の最大値が下がらないことが保証される。

エリートの遺伝子が集団の中で広まりすぎて、解の多様性が失われる。





# 交叉(crossover)

選ばれた 2 つの個体の遺伝子の一部を入れ替える操作。

生物の交配をモデル化したもの。

## 交叉確率( $P_c$ )

各々の染色体において、交叉が発生する確率。

通常、80%～95%として設定する。

# 基本的な交叉方式

一点交叉

(single point crossover)

二点交叉

(two-point crossover)

多点交叉

(multi-point crossover)

一様交叉

(uniform crossover)

## 一点交叉

(single point crossover)

## 二点交叉

(two-point crossover)

## 多点交叉

(multi-point crossover)

## 一様交叉

(uniform crossover)

## 一点交叉 (single point crossover)

遺伝子が交叉する場所(交叉点; crossover point)をランダムに決定し、その場所より後ろの遺伝子を入れ替える。

# 一点交叉

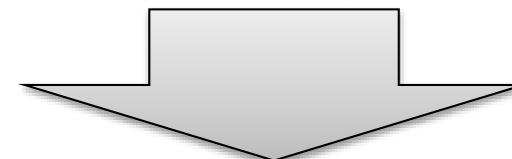
(single point crossover)

染色体1

1001011101101

染色体2

1010011010111



子孫1

1001011010111

子孫2

1010011101101

## 一点交叉 (single point crossover)

効率が低いため、現在はあまり用いられていない。

基本的な交叉

一点交叉

(single point crossover)

二点交叉

(two-point crossover)

多点交叉

(multi-point crossover)

一様交叉

(uniform crossover)

## 二点交叉 (two-point crossover)

2つの交叉点をランダムに決定し、  
その間の遺伝子を入れ替える。

## 二点交叉

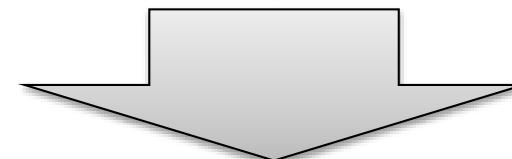
(two-point crossover)

染色体1

1001011101101

染色体2

1010011010111



子孫1

1001011011101

子孫2

1010011100111

一点交叉

(single point crossover)

二点交叉

(two-point crossover)

多点交叉

(multi-point crossover)

一様交叉

(uniform crossover)

## 多点交叉 (multi-point crossover)

3つ以上の交叉点をもつ方法。

二点交叉や一様交叉より良い値が得られることが殆どないため、あまり用いられていない。

一点交叉

(single point crossover)

二点交叉

(two-point crossover)

多点交叉

(multi-point crossover)

一様交叉

(uniform crossover)

## 一様交叉 (uniform crossover)

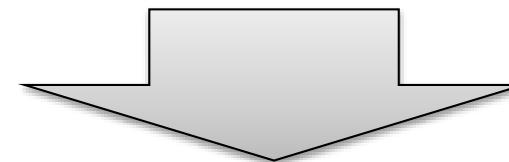
各遺伝子のそれぞれを所定の確率  
(通常は $1/2$ ) で入れ替える。

# 一様交叉 (uniform crossover)

染色体1

1001011101101101  
1010011010101111

染色体2



子孫1

10110110001101

子孫2

10000111101111

## 一様交叉 (uniform crossover)

2点交叉が得意とする問題を苦手とし、逆の性質を示す。

ヒッチハイキング問題への対策として用いられる。

順列エンコーディングに  
対応した交叉方式

同じ遺伝子が  
重複しないこと

循環交叉  
(cycle crossover; CX)

順序交叉  
(order crossover; OX)

一様順序交叉  
(order-based crossover; OX2)

一様位置交叉  
(position-based crossover; POX)

部分写像交叉

(partially mapped crossover; PMX)

サブツアー交換交叉

(subtour exchange crossover; SXX)

辺組換え交叉

(edge recombination crossover; ERX)

枝組立て交叉

(edge assembly crossover; EAX)

# 循環交叉 (cycle crossover; CX)

順序交叉  
(order crossover; OX)

一様順序交叉  
(order-based crossover; OX2)

一様位置交叉  
(position-based crossover; POX)

## 循環交叉 (cycle crossover; CX)

双方の親から重複しない組み合わせを選んで、遺伝子の一部ずつを受け継ぐ。

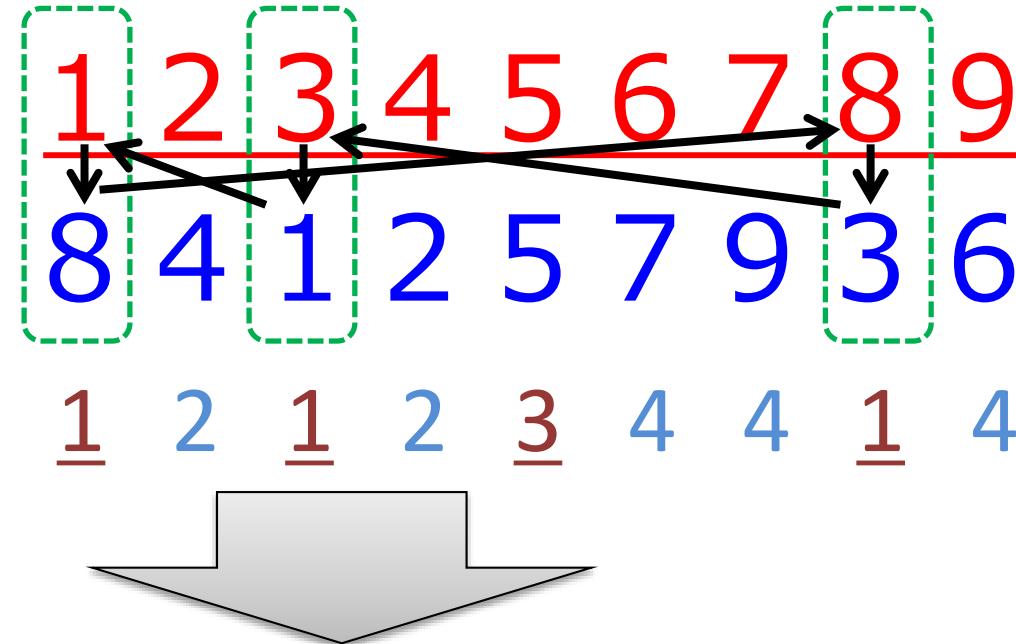
## 循環交叉

(cycle crossover; CX)

染色体1

染色体2

サイクル#



子孫1

1 4 3 2 5 7 9 8 6

子孫2

8 2 1 4 5 6 7 3 9

循環交叉  
(cycle crossover; CX)

順序交叉  
(order crossover; OX)

一様順序交叉  
(order-based crossover; OX2)

一様位置交叉  
(position-based crossover; POX)

## 順序交叉 (order crossover; OX)

片方の親からは一部をそのまま受け継ぎ、残りの部分についてはもう片方の親から相対的な順序を受け継ぐ。

## 順序交叉

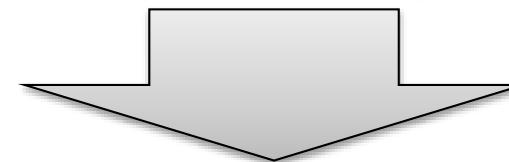
(order crossover; OX)

染色体1

1 2 3 4 5 6 7 8 9

染色体2

8 4 2 7 5 1 9 3 6



子孫1

1 2 3 4 8 7 5 9 6

子孫2

8 4 2 7 1 3 5 6 9

循環交叉  
(cycle crossover; CX)

順序交叉  
(order crossover; OX)

一様順序交叉  
(order-based crossover; OX2)

一様位置交叉  
(position-based crossover; POX)

## 一様順序交叉

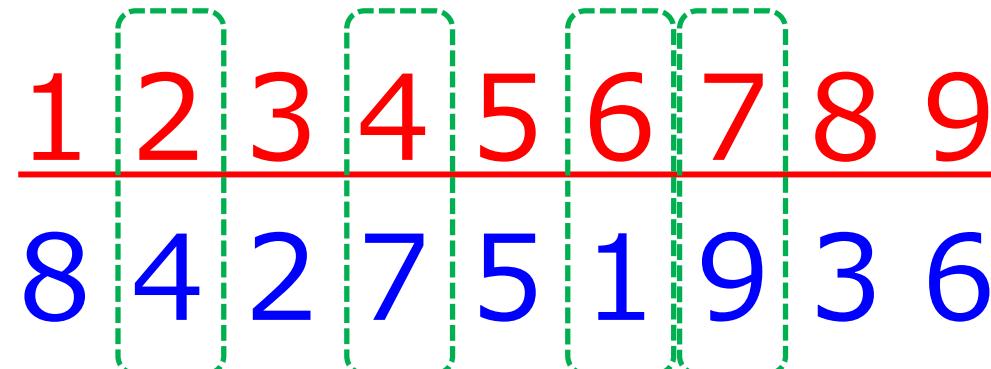
(order-based crossover; OX2)

一様に選んだ遺伝座について、一方の親の遺伝子の出現順序に従つて、他方の親の遺伝子を並べ替えたものを受け継ぐ。

## 一様順序交叉

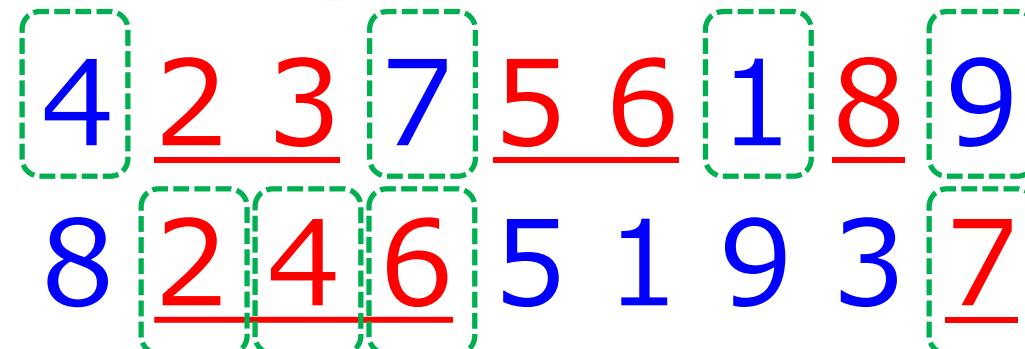
(order-based crossover; OX2)

染色体1



染色体2

子孫1



子孫2

循環交叉  
(cycle crossover; CX)

順序交叉  
(order crossover; OX)

一様順序交叉  
(order-based crossover; OX2)

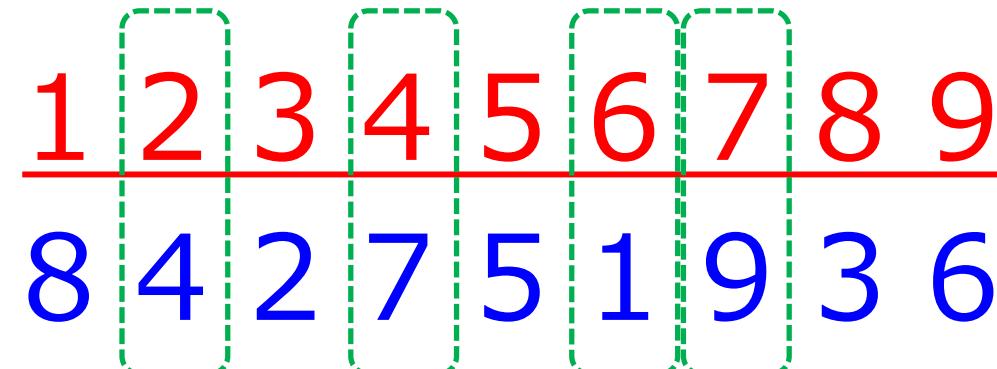
一様位置交叉  
(position-based crossover; POX)

## 一様位置交叉 (position-based crossover; POX)

一様に選んだ遺伝子を入れ替え、  
残りの遺伝子については親と同じ  
相対順序になるように配置する。

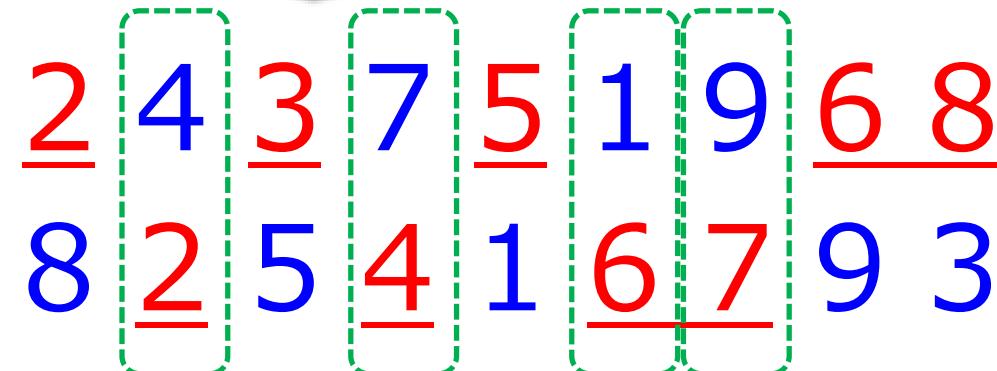
# 一様位置交叉 (position-based crossover; POX)

染色体1



染色体2

子孫1



子孫2

## 部分写像交叉

(partially mapped crossover; PMX)

サブツアー交換交叉

(subtour exchange crossover; SXX)

辺組換え交叉

(edge recombination crossover; ERX)

枝組立て交叉

(edge assembly crossover; EAX)

# 部分写像交叉

(partially mapped crossover; PMX)

片方の親 $P_1$ からは一部をそのまま受け継ぎ、残りの部分についてはもう片方の親 $P_2$ から重複しない遺伝子をそのまま受け継ぐ。

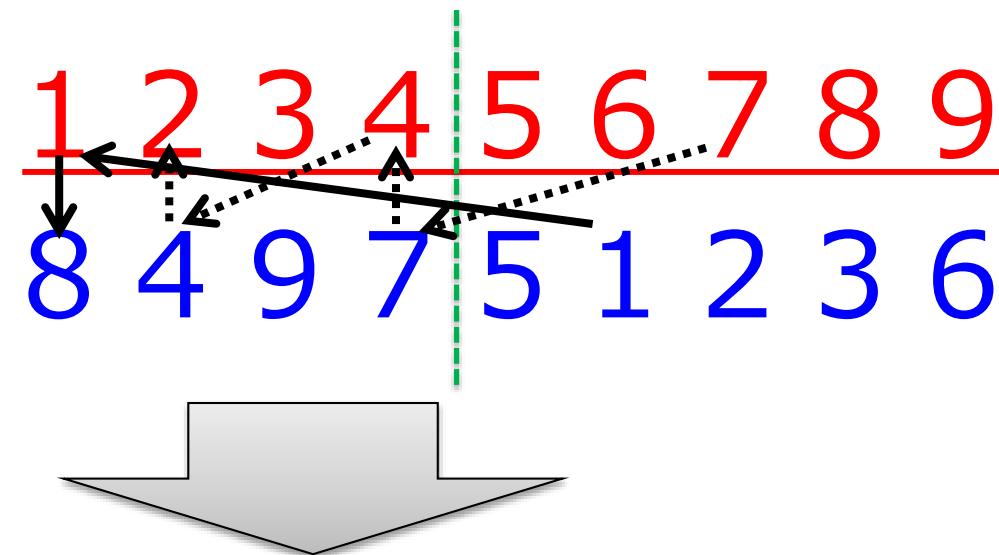
重複する遺伝子については、 $P_1$ 上で該当する遺伝子が占める遺伝子座を求め、 $P_2$ 上の同じ遺伝子座に存在する遺伝子を受け継ぐ。

## 部分写像交叉

(partially mapped crossover; PMX)

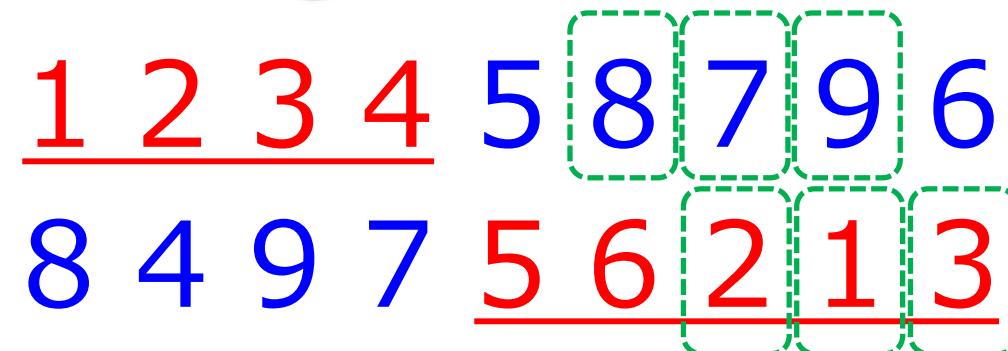
染色体1

染色体2



子孫1

子孫2



部分写像交叉

(partially mapped crossover; PMX)

サブツアー交換交叉

(subtour exchange crossover; SXX)

辺組換え交叉

(edge recombination crossover; ERX)

枝組立て交叉

(edge assembly crossover; EAX)

## サブツリー交換交叉

(subtour exchange crossover; SXX)

それぞれの親の間である共通する数字の組み合わせからなる部分順列が存在する場合に、それを交換する。さらに入れ替えた部分順列を逆にしたものを含め、合計 4 個の子を生成する。

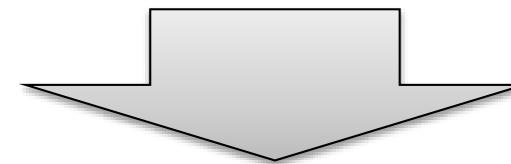
# サブツリー交換交叉 (subtour exchange crossover; SXX)

染色体1

1	2	3	4	5	6	7	8	9
---	---	---	---	---	---	---	---	---

染色体2

8	1	7	5	3	4	2	9	6
---	---	---	---	---	---	---	---	---



子孫1

1	5	3	4	2	6	7	8	9
---	---	---	---	---	---	---	---	---

子孫2

8	1	7	2	3	4	5	9	6
---	---	---	---	---	---	---	---	---

子孫3

1	2	4	3	5	6	7	8	9
---	---	---	---	---	---	---	---	---

子孫4

8	1	7	5	4	3	2	9	6
---	---	---	---	---	---	---	---	---

部分写像交叉

(partially mapped crossover; PMX)

サブツアー交換交叉

(subtour exchange crossover; SXX)

辺組換え交叉

(edge recombination crossover; ERX)

枝組立て交叉

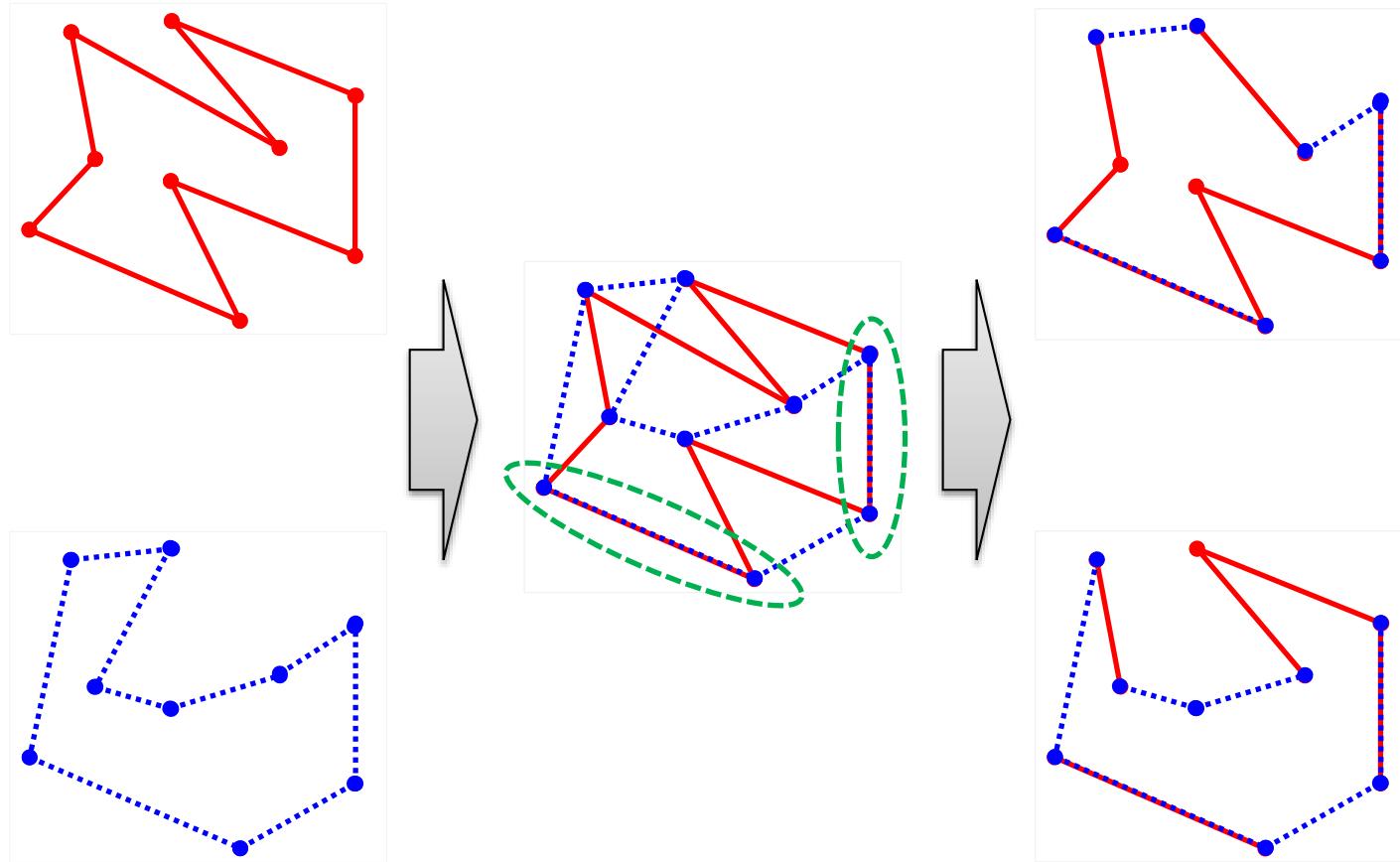
(edge assembly crossover; EAX)

## 辺組換え交叉

(edge recombination crossover; ERX)

順列の繋がり(辺)に着目した交叉方法。双方の親が共通して保有する辺をなるべく多く受け継ぐように子を生成する。

# 辺組換え交叉 (edge recombination crossover; ERX)



部分写像交叉

(partially mapped crossover; PMX)

サブツアー交換交叉

(subtour exchange crossover; SXX)

辺組換え交叉

(edge recombination crossover; ERX)

枝組立て交叉

(edge assembly crossover; EAX)

## 枝組立て交叉

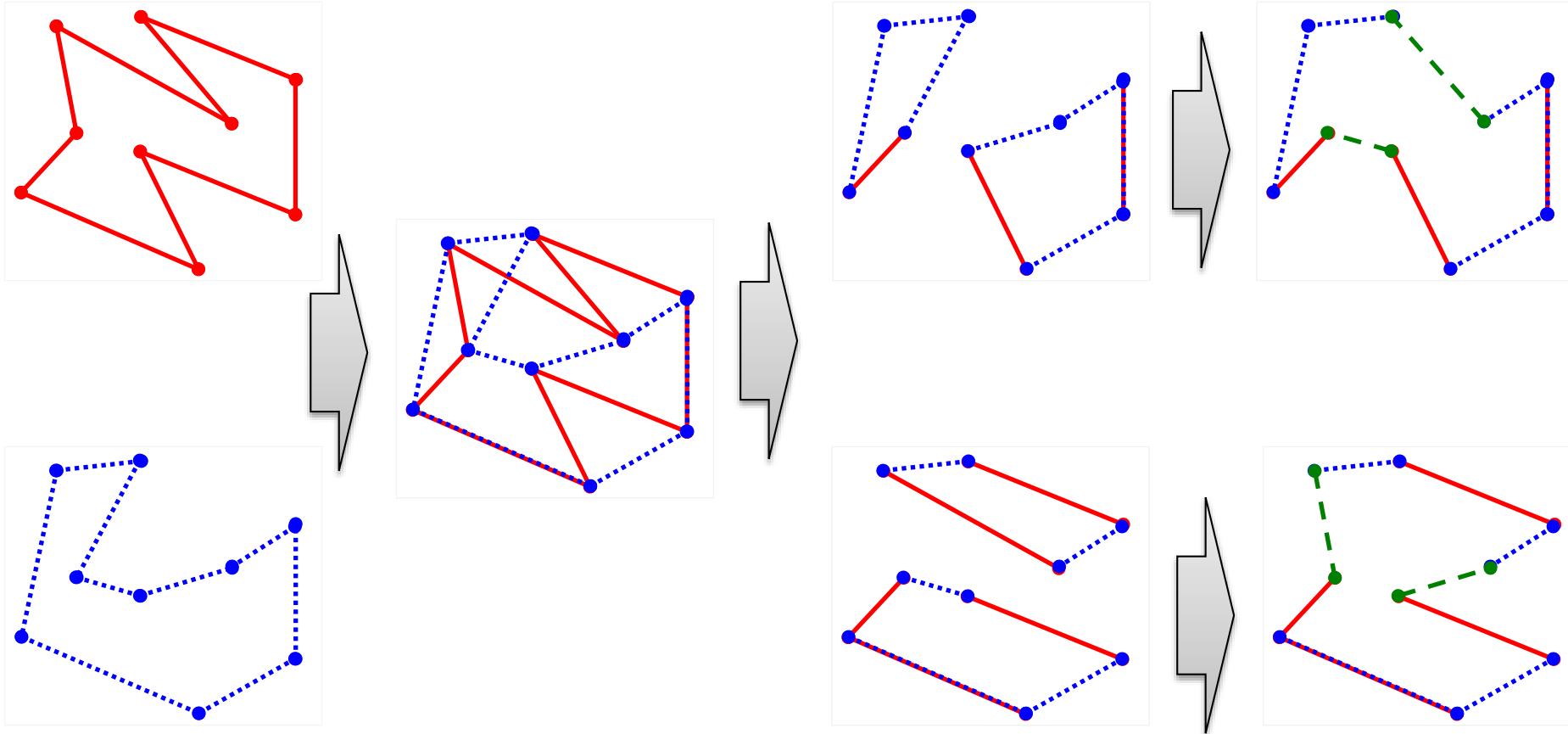
(edge assembly crossover; EAX)

両親のいずれかが保有する辺のみを用いて複数のサブツリーに分割された不完全解を生成する。

サブツリー間を、なるべく短い辺を用いて接続しなおすことで完全解を含む子を生成する。

# 枝組立て交叉

(edge assembly crossover; EAX)



## 枝組立て交叉

(edge assembly crossover; EAX)

経路問題への適用として、計算コストと精度の両面から、現時点において最も優れた方法とされてい  
る。

# その他の交叉方式

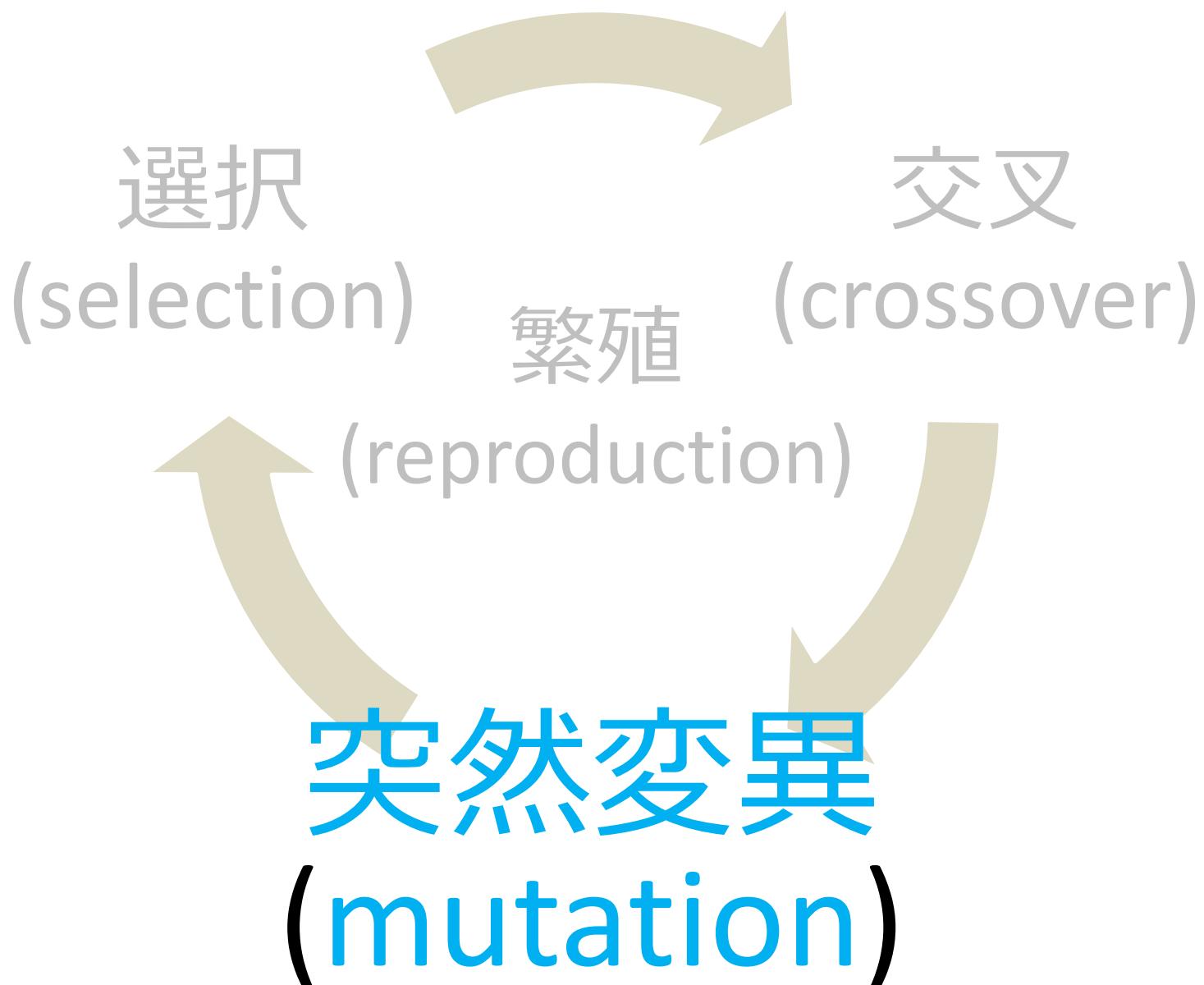
算術交叉  
(arithmetic crossover)

平均化交叉

ブレンド交叉(BLX- $\alpha$ )

单峰性正規分布交叉(UNDX)

シンプレックス交叉(SPX)



# 突然変異(mutation)

個体の遺伝子の一部をランダムに変化させる。

生物における突然変異をモデル化したもの。

局所解に陥ることを防ぐ効果がある。



# 変異確率(Pm)

各々の染色体において、突然変異が発生する確率。

通常、0.1%～1%、高くても数%として設定する。

確率が低すぎると局所解に陥りやすくなり、高すぎるとランダム探索に近づいて解が収束しにくくなる。



置換(substitution)

摂動(perturbation)

交換 swap

逆位(inversion)

搅拌(scramble)

転座(translocation)



**重複(duplication)**

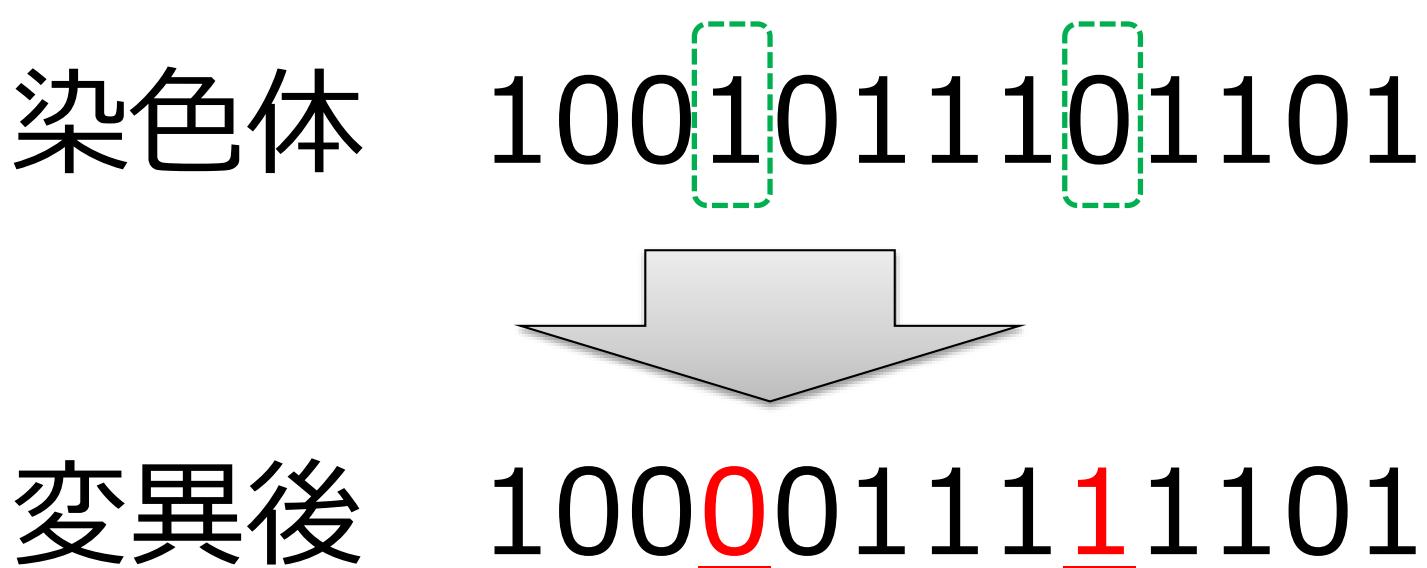
**挿入(insertion)**

**欠失(deficiency; deletion)**



# 置換(substitution)

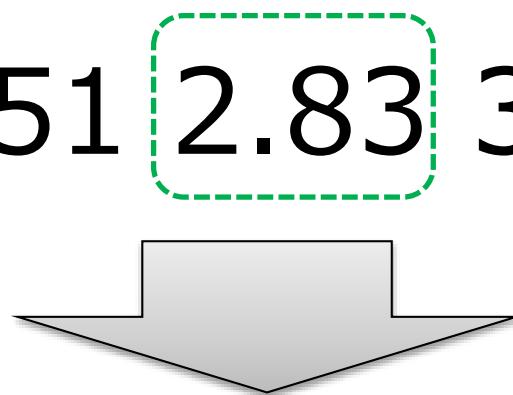
ランダムに選ばれた遺伝子を対立遺伝子に置き換える。



# 擾動(perturbation)

ランダムに選ばれた遺伝子の値に  
**微量値**を加算(または減算)する。

染色体 0.51 2.83 3.12 1.50



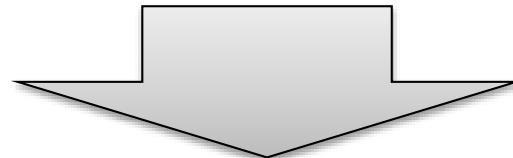
変異後 0.51 2.76 3.12 1.50



# 交換(swap)

ランダムに選ばれた 2 つの遺伝子を入れ替える。

染色体 1 2 3 4 5 6 7 8 9

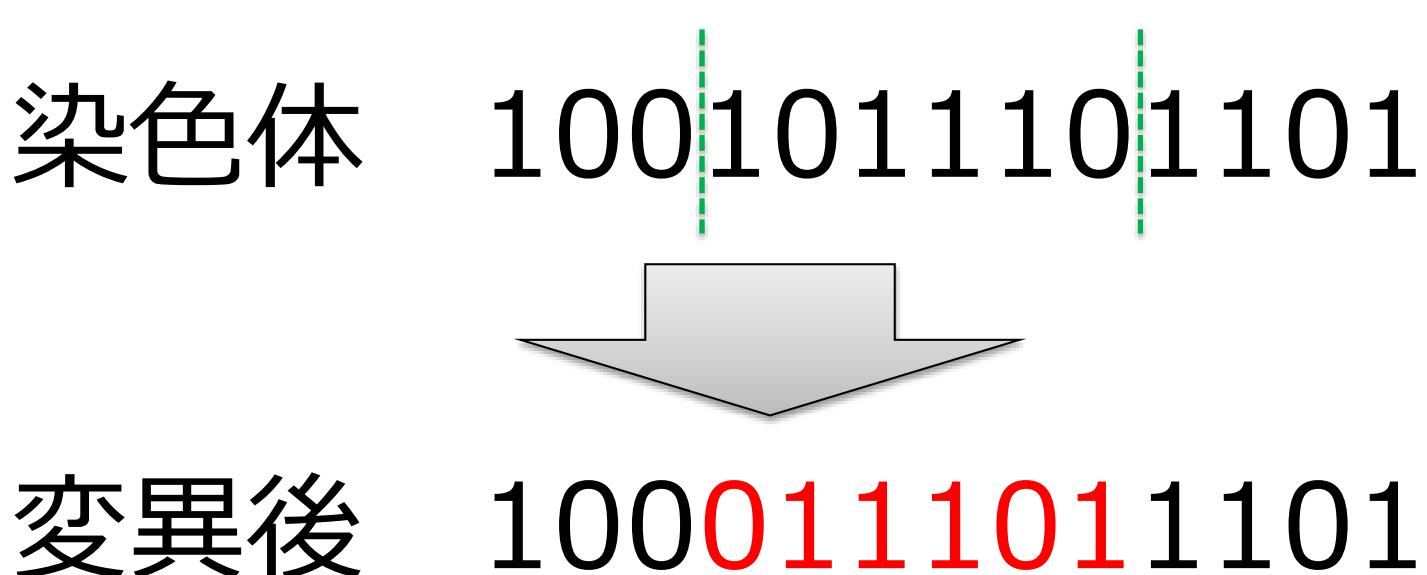


変異後 1 2 6 4 5 3 7 8 9



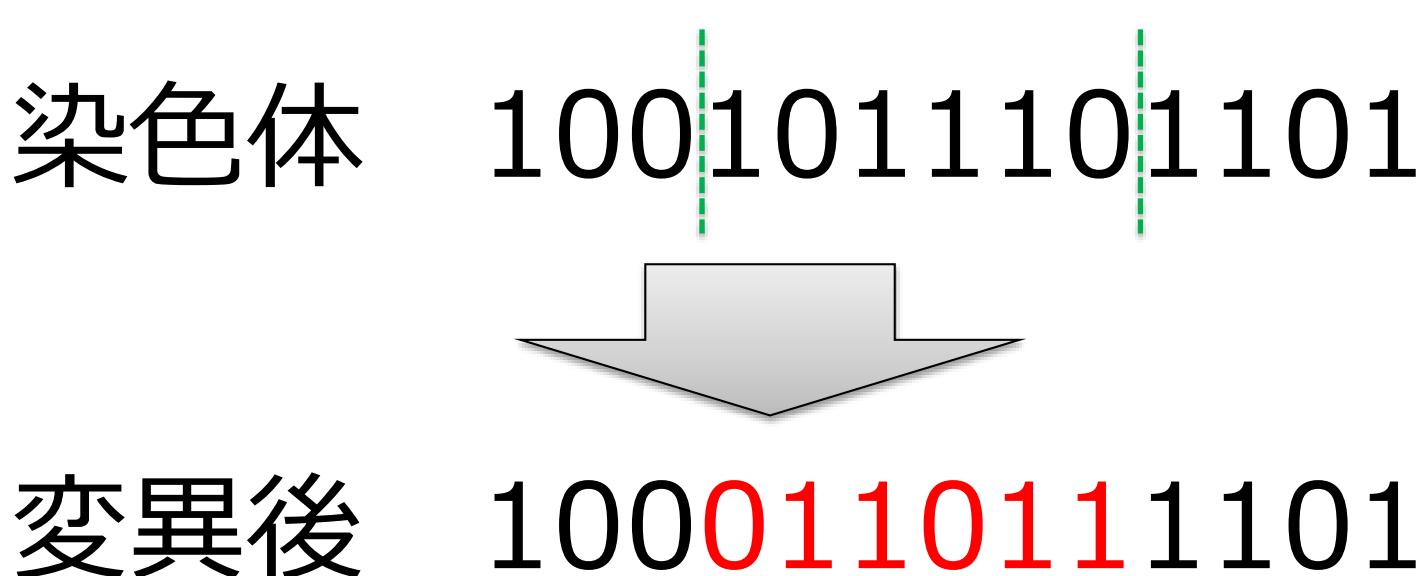
# 逆位(inversion)

ランダムに選ばれた2点間の遺伝子を逆順に並べ替える。



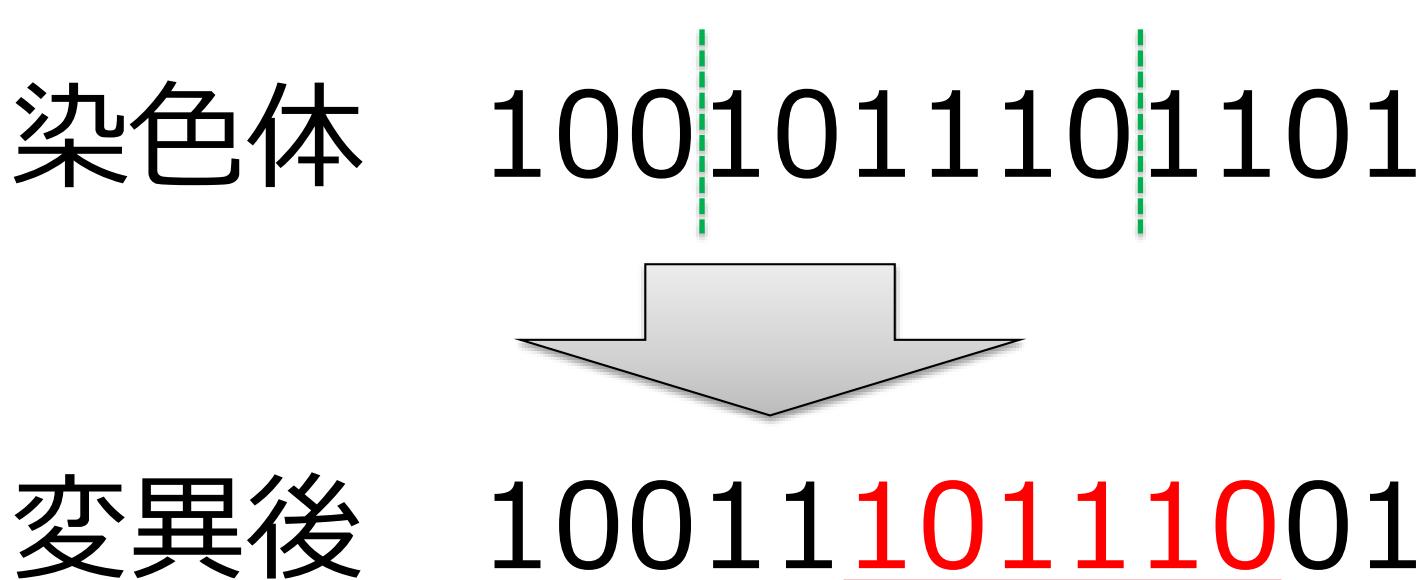
# 搅拌(scramble)

ランダムに選ばれた 2 点間の遺伝子をランダムに並べ替える。



# 転座(translocation)

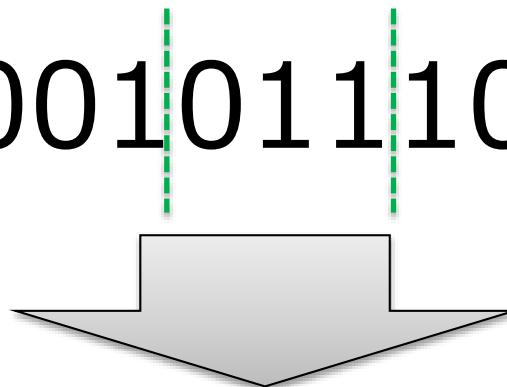
ランダムに選ばれた 2 点間の遺伝子を別の遺伝子座に移動する。



# 重複(duplication)

ランダムに選ばれた 2 点間の遺伝子を別の遺伝子座に複製する。

染色体 1001011|101101



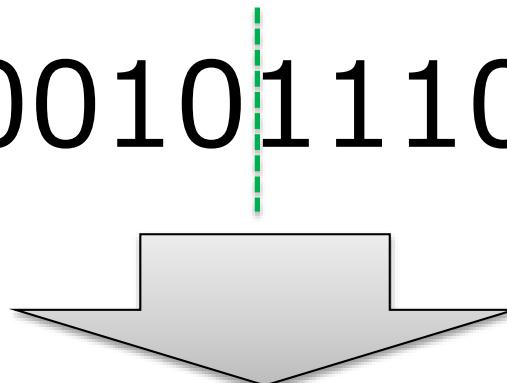
変異後 1011011101101101  
遺伝子長が変化



# 挿入(insertion)

ランダムに選ばれた場所に任意の遺伝子を挿入する。

染色体 10010|11101101



変異後 100100111101101

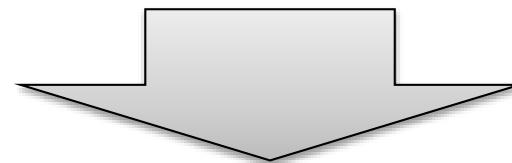
遺伝子長が変化



# 欠失(deficiency; deletion)

ランダムに選ばれた遺伝子を削除する。

染色体 1001011101101



変異後 1000111101

遺伝子長が変化



# New Generation Alternation Model (世代交代モデル)



離散世代モデル  
(discrete generation model)

連續世代モデル  
(continuous generation model)



離散世代モデル  
(discrete generation model)

連續世代モデル  
(continuous generation model)



# 離散世代モデル (discrete generation model)

全ての個体が一斉に繁殖を行い、現世代集合全体が次世代集合に置き換えられるモデル。

世代間における個体のオーバーラップが存在しない。



## 単純GAモデル (simple GA)

ルーレット選択と交叉・突然変異を用いて、親世代と同数の子個体を生成し、子個体のみを次世代として残す。

初期収束による局所解の発生や、進化的停滯が起こりやすい。



離散世代モデル  
(discrete generation model)

連續世代モデル  
(continuous generation model)



# 連續世代モデル (continuous generation model)

現在の世代のうち一定の割合の個体だけを入れ替え、のこりはそのまま次世代に残すモデル。



## 世代ギャップ (generation gap)

離散世代モデルにおいて、各世代の個体のうち次世代と入れ替えるものの割合。



## 定常状態モデル (steady state)

1 世代につき 1 組（2 個）の親個体から子個体を生成し、現世代のうち最も適応度の低い個体と入れ替える。



# 世代間最小ギャップモデル (minimum generation gap; elitist recombination)

ランダムに選択された親個体2個と、そこから作られる子個体2個を1つの「家族」とする。

家族の中でエリート選択とルーレット選択により、2個体を次世代に残す。これを家族の数だけ繰り返す。



## 世代間最小ギャップモデル (minimum generation gap; elitist recombination)

親子個体の選択を適応度を考慮せずにランダムに行うため、多様性維持に優れており、初期収束が起こりにくい。

家族内で選択を行うため、計算負荷が軽く、並列計算への適応もしやすい。



# Known Issues



# 初期収束

## ヒッチハイキング



# 初期収束

ヒッチハイキング



# 初期収束

最初の方の世代で「偶然」他の個体より適応度が圧倒的に高い個体が生じた場合、その個体の遺伝子が集団中に爆発的に増えて、探索がかなり早い段階で収束してしまう現象。



各パラメータを繰り返し設定し  
なおしてトライするしかない。



- 集団数を増やす
- ランキングの選択確率を変更する
- トーナメントサイズを縮小する
- 突然変異率を増やす
- 適用する問題にたいして効果的となるよう、突然変異操作を変更する



# 大変異

あらかじめ定めた条件に応じて、  
特定の世代においてのみ突然変  
異率を急激に高める方法。

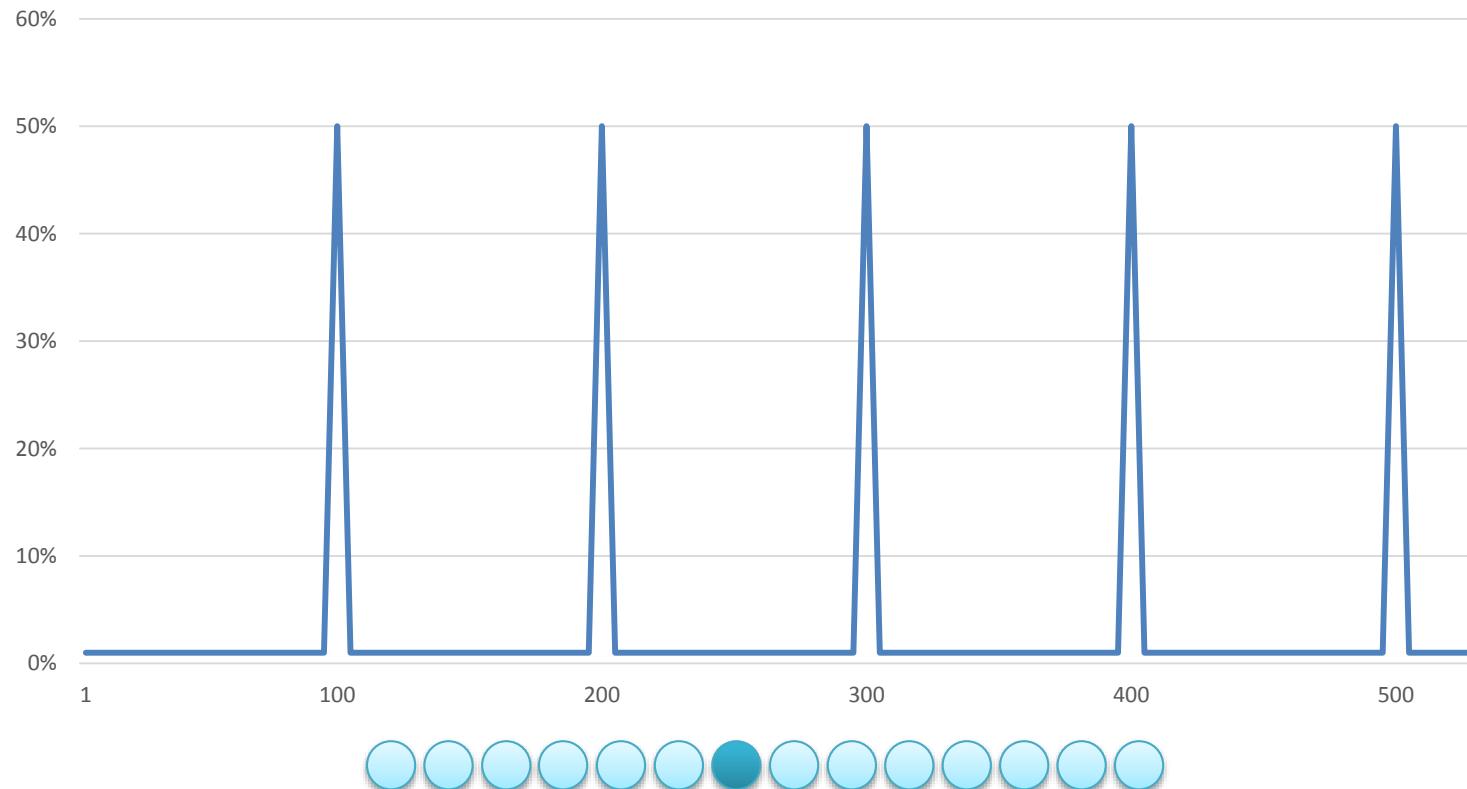
自然界における環境の激変（隕  
石の激突など）に相当する。



# 大変異

例えば...

通常 1 %に設定した突然変異率を、  
1 0 0で割り切れる世代のみ50%とする。



## 適応変異

親個体間の不一致度合い(ハミング距離)に応じて、生成された子個体の突然変異率を変化させる。



## 適応変異

例えば…

染色体1

1001011101101
1010011010111

染色体2

ハミング距離 = 6 → 突然変異率 = 1%

染色体1

1001011101101
1000011100111

染色体2

ハミング距離 = 2 → 突然変異率 = 8%



# 初期収束

## ヒッチハイキング



# ヒッチハイキング

遺伝子操作時に最適解と一致しない遺伝子が混入してしまうことによって最適解の生成を妨げる現象。  
2点交叉で発生しやすい。



最適解 = ...11110100000...

に対して、

親1 = ...11110000000...

親2 = ...11111110000...

が交叉して

最適解を得られる確率 $p$ は・・・



二点交叉の場合、

$$p = \frac{2}{L(L-1)}$$

( L = 遺伝子長 )

遺伝子長が増加するにつれて、  
加速度的に確率が低下する。



しかし、  
一様交叉であれば、

$$p = \frac{2}{2^3} = \frac{1}{4}$$

となり、遺伝子長に依存しない。



# Summary



GAとは、  
生物の遺伝を模したアルゴリズム。  
適者生存による自然淘汰で最適解  
を探索することができる。  
対象となる問題に関する数学的な  
知識がない場合でも適用が可能。



問題の性質に適したエンコーディングと交叉方法を選ぶことが重要。組み合わせの増加などにより指数関数的に複雑さが増加する問題に対しても、線形的な時間増加で探索を行うことができる。



局所最適解への収束を防ぐために、  
状態に応じて突然変異の方法や、  
確率、トーナメントサイズなどの  
パラメータを変えることも必要。



GAで  
いろいろな問題を  
解いてみよう！



# Genetic Algorithms Programming Library for JavaScript



<https://github.com/kzokm/ga.js>

## Samples of Genetic Algorithm



<http://ga-samples.herokuapp.com/>



<http://aitc.jp>



<https://www.facebook.com/aitc.jp>



ハルミン  
AITC非公式イメージキャラクター