平成 14 年度

学士学位論文

順列符号化 GA 開発環境の構築

Implementation of Development Environment for Genetic Algorithms with Permutation Encoding

1030234　　赤間 寛

指導教員　　坂本 明雄

2003 年 2 月 12 日

高知工科大学 情報システム工学科

要 旨

順列符号化 GA 開発環境の構築

赤間 寛

遺伝的アルゴリズム (Genetic Algorthm: GA) は主に初期解集団生成，選択淘汰，交叉，突然変異の 4 つの部分により成り立っている．これら 4 つの部分は，適用する問題に関わらず再利用可能なコアの部分といえる．

本研究では上記 4 つのコアとなる部分を汎用的に作成し，プログラム作成時にコアとなる部分は再利用することにより，開発時間の短縮をはかるものである．

また，巡回セールスマン問題 (Traveling Salesman Problem: TSP) を本環境に適用してプログラムを作成し，実験を行う．

キーワード　遺伝的アルゴリズム，順列符号化，巡回セールスマン問題

Abstract

Implementation of Development Environment for Genetic Algorithms with Permutation Encoding

Akama Hiroshi

Genetic Algorthm(GA) mainly consists of initialization, selection, crossover and mutation. These four portions are reusable cores regardless of each optimization problem to apply.

In this research, these four portions are implemented by C language to serve the cores at the time of program creation aims at shortening of development time by reusing.

This environment is applied to Traveling-Salesman-Problem(TSP), and the experimental results are shown.

key words Genetic Algorithm, Permutation Encoding, Traveling Salesman Problem

目次

図目次

図目次

表目次

第 1 章

序論

遺伝的アルゴリズム (Genetic Algorithm: GA) は，生物の遺伝と進化のメカニズムを模倣したアルゴリズムである．このアルゴリズムは，世代を形成している個体の集団 (解集団) の中で，環境に高い適応度を示す個体ならば，複製，交叉および突然変異を経て，その遺伝的特徴を受け継ぎ次世代を形成していき，世代を重ねるごとに環境における適応度を高めていく，最適化手法のひとつである．

GA を用いてプログラムを作成する場合，対象となる組合せ最適化問題の種類が違っても，GA の特性上同様の動作を行うコアとなる部分が存在する．例えば，選択，交叉，突然変異などがコアにあたる．つまり，コアとなる部分を予め汎用的に作成しておけば，実際の問題について GA を適用しプログラムを作成する場合は，問題固有の操作である適応度を求める計算部分のみを作成し，コアとなる部分は再利用することで開発時間の短縮ができると考えられる．

本研究では再利用可能な GA のコア部分を作成し，GA を利用するプログラムに組み込むことで開発時間の短縮をはかることを目的としている．

また，組み合せ最適化問題を解くために，GA を用いると，2 進数符号化による染色体表現では，問題の解を適切に表現することが困難である場合が多いことから，本研究で開発する環境は，順列表現によって符号化されている染色体を扱うことを前提としている．

本論文の構成は以下の通りである．次章では，遺伝的アルゴリズムについて説明し，3 章では本環境の構成，4 章では利用する上での注意点，5 章では本環境の適用例として組合せ最適化問題の一つである巡回セールスマン問題 (Traveling Salesman Probrem: TSP) を実装した場合，そして 6 章で結論を述べる．

第 2 章

遺伝的アルゴリズム

遺伝的アルゴリズム (Genetic Algorthm: GA) は，ダーウィンの進化論に着想を得た，生物の遺伝と進化のメカニズムを工学的にモデル化したアルゴリズムである [1]．すなわちある個体が現在の環境に適しているならば，他の個体より有利な特徴を持ち，その性質を次世代へと伝えられ，そうでないならば，次世代へと性質を伝えることが困難であるという自然淘汰の摂理に基づいている．

下記に本論文中で述べる GA に関する用語の説明を示す [1][2][3].

- 染色体

  数値や記号を一次元に並べ解として扱っている構造体.

- 遺伝子

  染色体を構成する記号や数値.

- 対立遺伝子

  遺伝子がとることのできる相異なる値.

- 遺伝子座

  染色体に存在する遺伝子の位置.

- 遺伝子型

  1 つの遺伝子座における遺伝子の組合せパターン.

- 表現型

  染色体によって定義される形質.

2.1　全体の流れ

GA の基本的操作には，次の 3 つが挙げられる．実際には図 2.1 に示すような流れで処理が実行される．

- 選択 (Selection)

  集団中での適応度分布に従い，次のステップで交叉を行う個体の生存分布を決定する．適応度分布に基づいているので，適応度の高い個体ほどより多くの子孫を残し易いようにする．

- 交叉 (Crossover)

  2 つの染色体間で，遺伝子を組み換え，新しい個体を発生させる．

- 突然変異 (Mutation)

  染色体のある部分の値を強制的に変化させる．

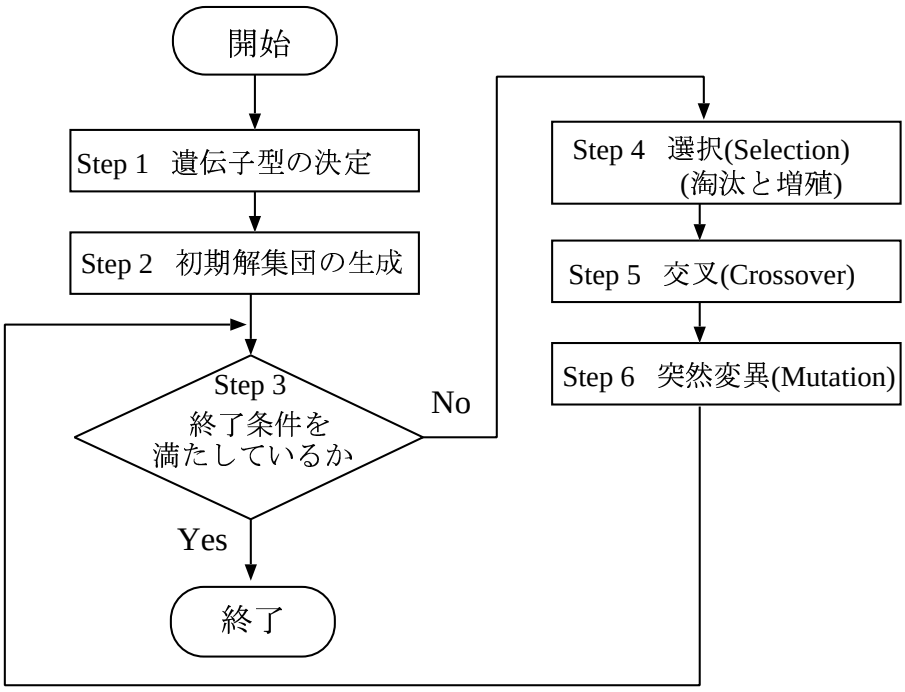

図 2.1　GA の流れ

2.1 全体の流れ

- Step1:遺伝子型の決定

  遺伝的アルゴリズムでは，対象となる問題を遺伝子の形で表現する必要がある．すなわち，どのように符号化するかを決定する必要がある．

- Step2:初期解集団の生成

  Step1 により決定した遺伝子型で符号化した，要素の異なる様々な染色体を発生させる．並列的な処理を特徴とする遺伝的アルゴリズムの特性上，発生させる個体数，すなわち集団のサイズが小さすぎる場合や，大きすぎる場合では十分な性能が発揮出来ないため，適当なサイズを決定する必要がある．

- Step3:終了条件のチェック

  このステップで考えられる終了条件とは，集団内の個体の適応度が基準値を満足する，または定められた世代数を経過しても集団内の最良個体の適応度が改善されない，以上の 2 点が考えられる．すなわち，前述の 2 点のうちのいずれか一方の終了条件を満たしたならば，終了し，そうでなければ次のステップに移る．どちらの終了条件の場合でも，適応度の計算をする必要がある．

- Step4:選択

  Step3 において求めた適応度に基づき，次のステップで交叉を行う個体の生存分布を決定する．適応度の低い個体の特徴は後世へ残すことができず，逆に適応度の高い個体の特徴は後世へ残すことができることを考慮すると，このステップは淘汰と増殖のステップとも考えられる．

- Step5:交叉

  2 つの染色体 (親) 間で遺伝子を組み換えて，新しい個体 (子) を発生させる．子は親の特徴を継承する．

- Step6:突然変異

  染色体中のある部分を強制的に変化させて，解集団の多様性を向上させる．この行為により，より良い適応度を持つ個体の発生が期待できる反面，染色体中の特徴が破壊されることもあり得る．

- Step7:繰返し

  Step3 に戻り，各個体の適応度を評価する．

2.2　符号化

組合せ最適化問題を GA を用いて解く場合，解を染色体 (符号化された記号列) の形に変換しなくてはならない．このとき，あまりにも簡素化された染色体へ符号化すると，解としての情報が欠落してしまう可能性が生じる．その反面あまりにも付加価値の多い符号化によって染色体を表現すると，適応度の計算に時間がかかってしまい，満足できる時間内に結果を算出するのが難しくなる．一般的にどのように解を符号化して，表現するかは経験的要素が大きい．

2.2.1　順列符号化

対象とする問題を解くために GA を用いる場合は，解を染色体の形に符号化しなくてはならない．なぜならば，GA においてその操作対象は染色体だからである．本研究では，順列符号化された染色体を操作対象とする GA 開発環境の構築を目的としているため，順列符号化することが有意義な問題に対して用いられることが前提である．ここで，順列符号化について説明する．以下に 2 進数符号化された染色体と順列符号化された染色体の一例を図 2.2 に示す．

Binary coding:　　0 1 1 1 0 1 0 0 1 0

Permutation coding:　9 2 0 5 3 6 1 4 7 8

図 2.2　染色体の 2 進数符号化と順列符号化

2 進数符号化では対立遺伝子が 0 と 1 であるのに対し，順列符号化では染色体長だけ対立遺伝子の種類が存在する．その理由として，順列符号化においては各々の遺伝子の重複が許

されないからである．順列符号化された染色体において，重複する遺伝子を含むことを致死遺伝子という．すなわち，問題の解として適当ではないことを意味する．また，2 進数符号化された染色体での交叉は切断点の数が 1 点，2 点を問わず注意する点はないが，順列符号化された染色体での交叉は致死遺伝子が発生しないように留意する必要がある．

2.3　適応度

問題の解候補を表現する染色体の適応度は，適応度関数 f により一意に決定される．GA は，高い適応度値をもつ個体が生き残るという自然淘汰の摂理と結びつけて，自然界の生殖過程をシミュレートする．すなわち世代が進むにつれて高い適応度値をもつ，より良い個体が発生することが期待されるように，適応度を最大化することを目的としている．

最小化問題を取り扱う場合，適応度値をもとに操作を行うと期待する結果が得られない．そこで，f を最大化すると値を最小化にするような評価関数 c への写像を考えなければならない．

2.4　選択

2.3 節により算出された，集団中の個体の適応度 (あるいは評価値) をもとに，解空間を構成する解集団から，交叉オペレータを行う対象である 1 対の親を確率的に選択する．よりよい子孫を残し適応度を高めるために，低い適応度をもつ個体よりも高い適応度をもつ個体の方が，選択される可能性を高くしなければならない．

2.5　交叉

交叉は 2.4 節の選択によって選ばれた，1 対の染色体 (親) から新たに染色体 (子) を生成する，遺伝的オペレータである．子は親の特徴を受け継ぎ，新たな解集団を形成する 1 要素となる．

2.6　突然変異

突然変異は染色体中のランダムに選ばれた遺伝子について，他の対立遺伝子に変更する．突然変異を行う理由は，親の世代で持ち合わせていない新しい特徴を導入することであり，その結果解集団に多様性がでてくる．しかし，突然変異が頻繁に発生すると，染色体中の特徴を破壊してしまう恐れがある．

2.7　淘汰

集団のサイズを一定に保つため，交叉によって生成された染色体 N 個と現世代の解集団のサイズ M の総計 $S = (N + M)$ から N 個の個体を淘汰しなくてはならない．貪欲な淘汰方法を採用する場合，適応度の低い順番に N 個を淘汰する．各世代毎に親と子を入れ替える場合，総計 S から M 個の個体を淘汰する．

第 3 章

本環境の説明

本章では本環境を構成している各種関数の説明を行う.

本環境は，初期解集団生成，選択，交叉，突然変異の 4 つの主要部分により構成されており，必要なパラメータも詳細に設定できる.

3.1　初期解集団生成

gene **fst_set(int n, int l)[*1]

初期解集団は染色体数 n と染色体長 l の 2 つのパラメータにより生成される．例えば，n=5, l=7 とした場合図 3.1 の初期解集団が生成される．なお，集団中の染色体はランダムに生成されているため，必ずしもこのパラメータで図 3.1 の初期解集団が生成されるわけではない.

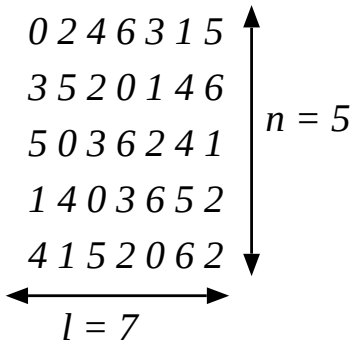

図 3.1　初期解集団

[*1] gene 型とは int 型を typedef を使用して別の型を付与をつけたものである．実際の動作は int 型と同等である.

3.2　選択

Pare *choice_pare(double *estimation，char option)[*2]

両親の選択は集団中の染色体の適応度 (評価値) を基に，指定された選択方法に基づき，集団内から複数の親のペアを選択する．

本研究で構築した環境では，両親選択方法にはルーレット戦略:r とトーナメント選択:t のどちらかを設定することができる．

3.2.1　ルーレット盤戦略

Pare *roulette(double *estimation)

ルーレット盤戦略とは，図 3.2 のように集団のそれぞれの個体に対して円盤上にその相対適応度に比例する面積の扇形領域を割り当て，円盤を回転させた後，停止したときに指し示す領域に対応する親を選択する方法である．

本環境での実装方法は，適応度の総和に対する各適応度の割合を算出し，乱数を用いて 0 から 1 までを発生させ，ランダムな値が各適応度の範囲内ならば，該当する親を選択するという方法を用いている．すなわち，適応度が低い個体でも有限の小さい確率により，親として選択される可能性があることを示唆している [2]．

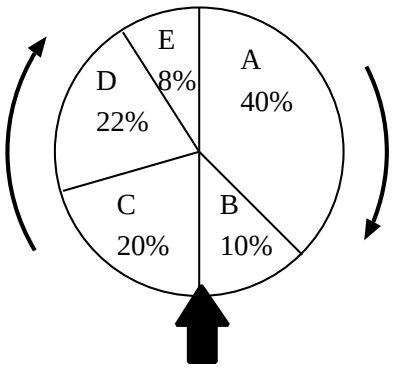

図 3.2　ルーレット盤戦略

[*2] Pare 型とは選択された親情報を記憶した構造体である．

3.2.2　トーナメント選択

Pare *tournament(double *estimation)

トーナメント選択とは，図 3.3 のように集団に含まれる染色体の間で，比較する選択法である．集団が M 本からなる染色体とし，集団を G 本 $(G \geq 2)$ の染色体からなる部分グループに分割し，それぞれのグループ内で最も高い適応度をもつ染色体を親の候補とする．そして，リスト内の要素を並べ換え，M 本の親が選択されるまで繰り返す.

本環境での実装方法は，G の値を $n/2$ とし，一度の選択で 1 対の親を選択するようにしてある．トーナメント選択の特色は，適応度が最良の染色体が G 回選択され，適応度が最悪の染色体は全く選択されないことである．

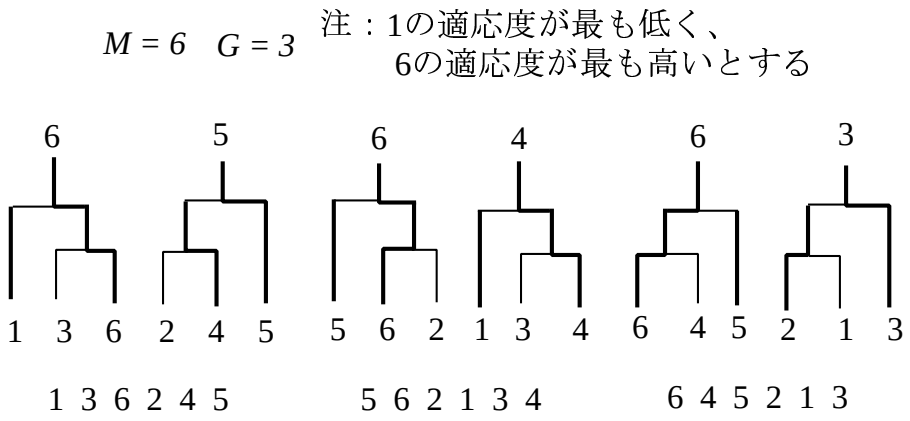

図 3.3　トーナメント選択

3.3　交叉

gene **crossover(Pare *pare，gene **set，char *xo_name)

3.2 節にて選択された 1 対の親に対し，交叉を行い次世代の解集団を形成する子孫を生成する．本環境では順列表現された染色体に操作を適用するため，単純交叉 (2 つの染色体の交叉部位の交換) は使用できない．なぜなら単純交叉により形成される染色体では，対立遺伝子が重複してしまう可能性があり，その結果，致死遺伝子が発生し，正しい解を表現でき

なくなるからである．

本環境で使用できる交叉方法は，解の順列表現に適している順序交叉，部分写像交叉，サイクル交叉，一様順序交叉の計 4 種類である．

順序交叉と部分写像交叉においては，切断点が必要である．切断点が 1 箇所の場合，交叉箇所が右 (1-point-right) か左 (1-point-left) かの 2 通り，切断点が 2 箇所の場合，交叉箇所が内側 (2-point-inside) か外側 (2-point-outside) かの 2 通り，すなわち計 4 通りの交叉方法がそれぞれ存在する．切断点 3 箇所以上も考えられるが，親から受け継ぐ染色体中の特徴を破壊してしまう恐れがあるため，本環境においては切断点の個数は 1 箇所，ないしは 2 箇所までと設定している．

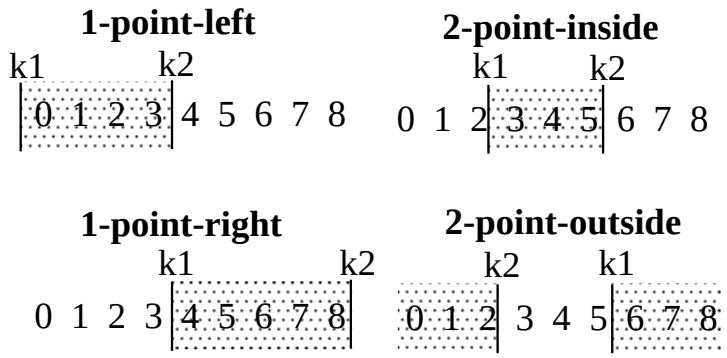

図 3.4　切断点

適用される切断点の値，範囲は以下の通りである．l は染色体長である [4].

- 1-point-right

$$1 \leq k1 \leq l-1, \quad k2 = l$$

- 1-point-left

$$k1 = 0, \quad 1 \leq k2 \leq l-1$$

- 2-point-inside

$$1 \leq k1 \leq l-1, \quad 1 \leq k2 \leq l-1, \quad k1 \leq k2$$

- 2-point-outside

$$1 \leq k1 \leq l-1, \quad 1 \leq k2 \leq l-1, \quad k2 \leq k1$$

以下，本章において図中で表す＊は切断点を意味する．また，交叉箇所とは図 3.5 において，網かけされている部分である．すなわち，1-point-left においては切断点の左側，2-point-inside においては内側が交叉箇所となる．

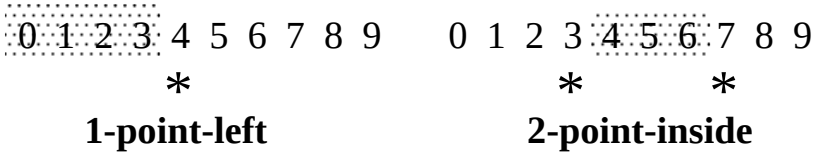

図 3.5　交叉箇所

交叉を実行するかどうかは，実行時に与えられる交叉確率 P_x によって判定される．P_x を閾値とし，乱数を発生させ交叉の実行を判定する．

本環境での交叉方法の指定には，xo_name に交叉種類の名前を文字列形式で与えることで各種交叉方法を指定することができる．交叉方法は，ox-1-r, ox-1-l, ox-2-in, ox-2-out, pmx-1-r, pmx-1-l, pmx-2-in, pmx-2-out, cx, uox の計 10 種類存在する．

それぞれの交叉方法では第 2 引数で与えられる解集団 set から，第 1 引数で与えられる親情報 pare をもとに 1 対の子を生成する．OX と PMX においては，第 3，4 引数で与えられる切断点 k1，k2 が必要である．

3.3.1　順序交叉 (Order Crossover: OX)

gene *ox(Pare *pare，gene **set，int k1，int k2)

OX は対立遺伝子の相対位置をもとに交叉を行う．図 3.6 から図 3.9 まで OX の交叉例をのせる．OX の基本的なアルゴリズムは以下の通りである．

1. 親 1 の交叉箇所以外を，子 1 に継承する．
2. 親 1 の交叉箇所の遺伝子の順序を，親 2 に現れる順序に並べ換え，継承する．

例えば，図 3.6 に示す OX-1-point-right の場合，親 1 の切断点の左側部分はそのまま子 1 へ継承し次に，親 1 の切断点の右側部分は，親 2 に遺伝子が現れる順序に従って並べ換え

を行い，子 1 の右側に継承する．子 2 の場合も同様の手順を行うことにより生成される．

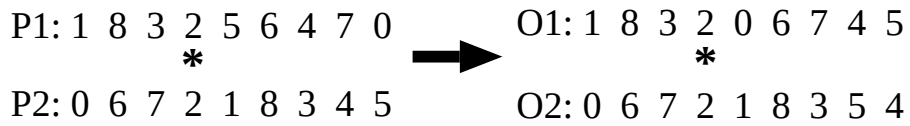

図 3.6　OX-1-point-right

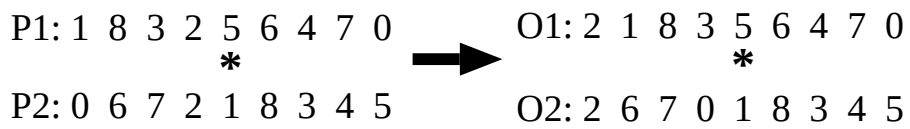

図 3.7　OX-1-point-left

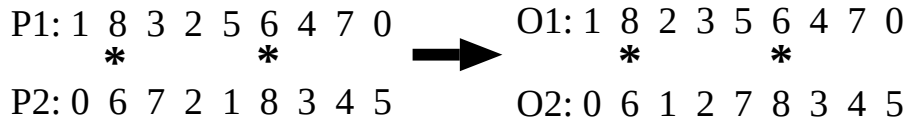

図 3.8　OX-2-point-inside

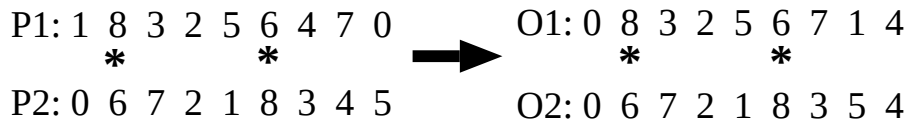

図 3.9　OX-2-point-outside

3.3.2　部分写像交叉 (Partially Mapped Crossover: PMX)

gene *pmx(Pare *pare，gene **set，int k1，int k2)

PMX は親 1 と親 2 の間で切断点の前後を交換し，致死遺伝子が発生しないように部分的にパッチを当てる交叉方法である．図 3.10 から図 3.13 まで PMX の交叉例をのせる．PMX の基本的なアルゴリズムは以下の通りである．

1. 親 2 の交叉箇所以外を，子 1 に継承する．
2. 親 1 の交叉箇所が，1 で子 1 が継承した遺伝子と重複していなければそのまま継承し，

重複していれば親 1 の親 2 に対する写像をあて，継承する．

例えば，図 3.10 に示す PMX-1-point-right の場合，まず親 2 の切断点の左側部分を子 1 にそのまま継承する．次に親 1 の 5 番目の遺伝子 ‘2’ を子 1 に継承するが既に子 1 に ‘2’ が存在し，致死遺伝子となるため継承できない．そこで，‘2’ の親 2 に対する写像である ‘0’ を継承するが，これも致死遺伝子となるので，さらに ‘0’ の親 2 に対する写像をたどり，‘8’ を子 1 に継承する．以下同じように重複がなければそのまま子 1 に継承し，重複があれば親 2 に対する写像をあて，致死遺伝子が発生しないように交叉を行う．子 2 に対しても同様の操作を行う．

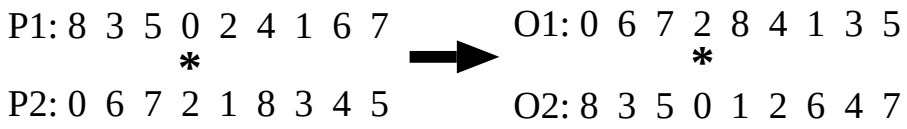

図 3.10　PMX-1-point-right

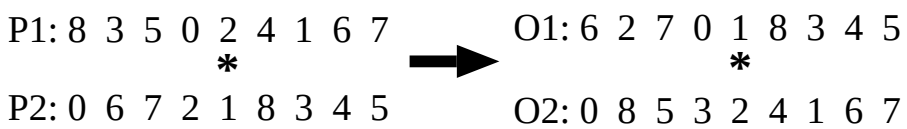

図 3.11　PMX-1-point-left

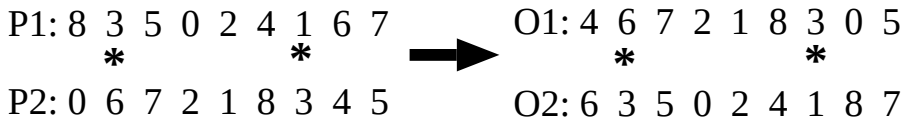

図 3.12　PMX-2-point-inside

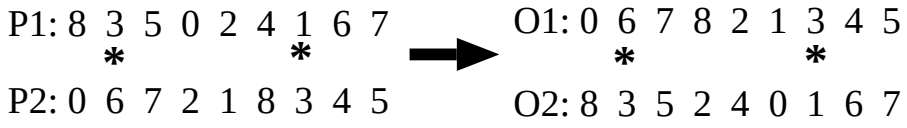

図 3.13　PMX-2-point-outside

3.3.3　サイクル交叉 (Cycle Crossover: CX)

gene *cx(Pare *pare，gene **set)

CX は，1 対の親の遺伝子座の位置が一致する，対立遺伝子からなる部分集合を含むサイクルを求め，子の同じ遺伝子座に継承する．

図 3.14 に示すように，親 1 の遺伝子座の位置 1 における遺伝子から始め，親 2 への写像を辿り，$\{1 \to 0 \to 5 \to 1\}$ というサイクルを得る．このサイクルを子 1 へ継承する．次に先程のサイクルに含まれていない遺伝子座に対して親 2 からサイクルを始め，$\{6 \to 8 \to 6\}$ というサイクルを得，子 1 に継承する．この操作をサイクルが得られるなくなるまで，繰り返す．子 2 に関しては始めのサイクルを親 2 から始めることで生成できる．

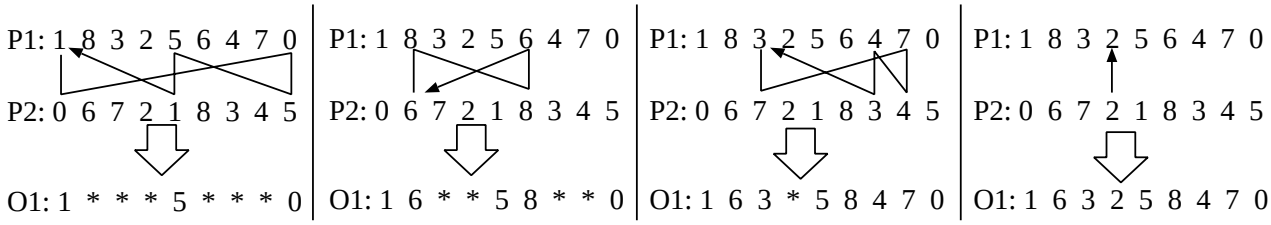

図 3.14　Cycle Crossover

3.3.4　一様順序交叉 (Uniformed Order Crossover: UOX)

gene *uox(Pare *pare，gene **set)

UOX は，2 進数記号列のテンプレートをもとに順序交叉を行う．テンプレートにおける 1 は親 1 から遺伝子を継承し，0 は，染色体中に現れる順序を保存して親 2 から遺伝子を継承する．

図 3.15 に示すように，親 1 からテンプレートの 1 の遺伝子座を示す {8，3，5，0} が子 1 に継承され，次にテンプレートの 0 の遺伝子座の遺伝子を親 1 から取り出し，親 2 に出現する順序に並べ換え，子 1 に継承させる．すなわち，この場合では，{1，2，6，4，7} が親 2 での出現順序により {6，7，2，4，1} となる．子 2 に関してはテンプレートの 0 と 1 の扱

いを逆にすることで生成できる．

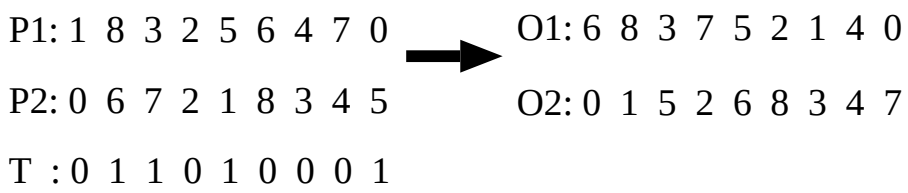

図 3.15　Uniformed Order Crossover

3.4　突然変異

gene *mutation(gene *chromosome，char *mutation_name)

解集団に交叉が適用された次の段階で，突然変異を適用するかどうかがチェックされる．突然変異が適用されると，第 1 引数で与えられる解集団中の適当な染色体に対し，操作を行う．突然変異を実行するかどうかは，実行時に与えられる突然変異確率 P_m によって判定され，このとき P_m の値を閾値としてそのまま採用するのではなく，$n \times P_m$ 回突然変異を実行することを保証するという形式をとっている．小数点以下の端数については，その値を閾値とし乱数を発生させ突然変異を実行するかどうかを判定する．突然変異を行う対象は解集団中のランダムな個体である．

本環境で使用できる突然変異は，ランダムな 2 点の交換 (two_mutate)，ランダムな 2 点間逆位 (rev_mutate)，ランダムな 2 点間を撹拌 (rand_mutate) の計 3 種類があり，mutation_name に突然変異の種類を文字列形式で与えることで，どの突然変異を用いるのかを指定することができる．

3.4.1　2 点交換

gene *two_mutate(gene *chromosome)

2 点交換は，図 3.16 に示すように染色体中のランダムに選ばれた 2 点を交換する，もっとも単純な突然変異である．

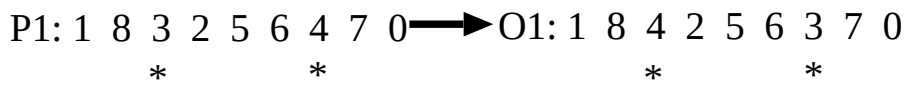

図 3.16　2 点交換

3.4.2　2 点間逆位

gene *rev_mutate(gene *chromosome)

2 点間逆位は，図 3.17 に示すように染色体中のランダムに選ばれた 2 点間において，現在の順序を逆順にする突然変異である．

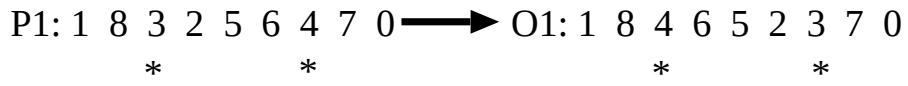

図 3.17　2 点間逆位

3.4.3　2 点間撹拌

gene *rand_mutate(gene *chromosome)

2 点間撹拌は，図 3.18 に示すように染色体中のランダムに選ばれた 2 点間において，ランダムに撹拌する突然変異である．

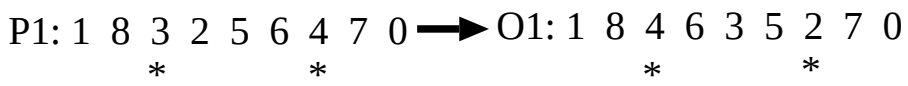

図 3.18　2 点間撹拌

第 4 章

本環境を利用する場合

本章では本環境を利用し，GA のプログラムを作成する際に，別途作成が必要である箇所についての説明をする．

4.1 適応度

集団内の個体 (解) を数値により状態を表した値が適応度である．GA はこの値をもとに各種操作を行っていく．

適応度の計算には，問題によって計算方法が違うため本環境では用意していない．ただし，本環境の他の部分との整合性に関係することから，適応度計算部分での出力は配列形式で行われなければならない．

4.2 評価値

GA は基本的に，適応度を最大化するように動作する．最適化問題において，目的が最大の評価を得るのであれば，いっこうに構わない．だが，最小化を目的とする問題の場合，適応度の大小関係を逆転する必要がある．すなわち，適応度を評価値へ写像し，適応度を必要とする関数で，評価値を採用すればよいといえる．

4.3　問題特有の関数

本環境で提供し得る関数は，全て GA の操作に必要なものであるが，GA を問題に適用する場合，必要になってくる部分があると考えられる．それが問題特有の関数である．

例えば，次章で説明する TSP については都市情報の読込みなどがそれにあたる．

第 5 章

巡回セールスマン問題への適用例

本章では本環境を TSP へ適用した場合について説明する．

巡回セールスマン問題 (Traveling Salesman Problem: TSP) とは N 個の頂点からなる完全グラフにおいて，各頂点が都市をあらわし，$D = [d_{ij}]$ の距離行列 (d_{ij} は都市 i と都市 j 間の距離を表す) をもとに，全ての都市を重複することなく巡回し，かつ都市間の距離の合計が最小になるような経路を求める問題であり，問題のサイズの多項式時間で最適解を求めるアルゴリズムが存在しないだろうと言われている，NP-困難な問題の 1 つである [2]

図 5.1 は本環境を TSP へ適用したアルゴリズムである．パラメータ設定で与えられるパラメータに集団サイズ n，未更新限度数 T_{limit} である．終了条件はカウンタの値が T_{limit} をこえた場合である．カウンタの値は，終了条件のチェックを行う毎に増えていき，現在の最良解が更新された時点で値を 0 にリセットする．

ここで，図中の 2 重線で囲まれている箇所は，新たに作成した部分である．それ以外は本環境によって構成されている．

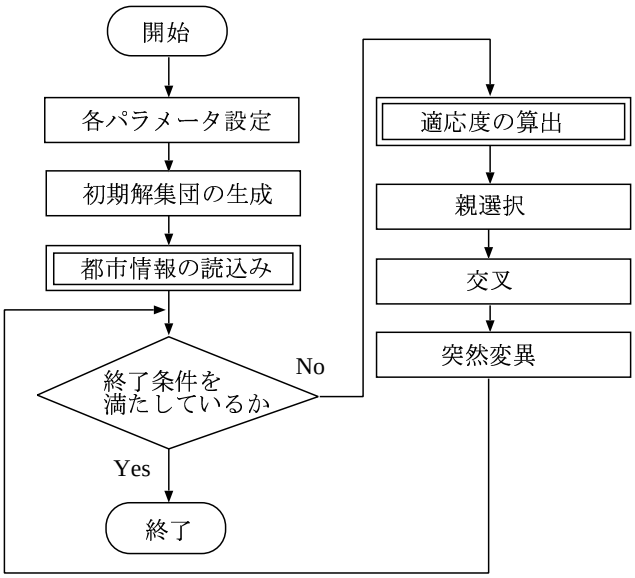

図 5.1　TSP アルゴリズム

例えば図 5.2 のような都市図が与えられた場合，$A \to C \to E \to D \to B \to A$ が距離 15 で最短ルートとなる.

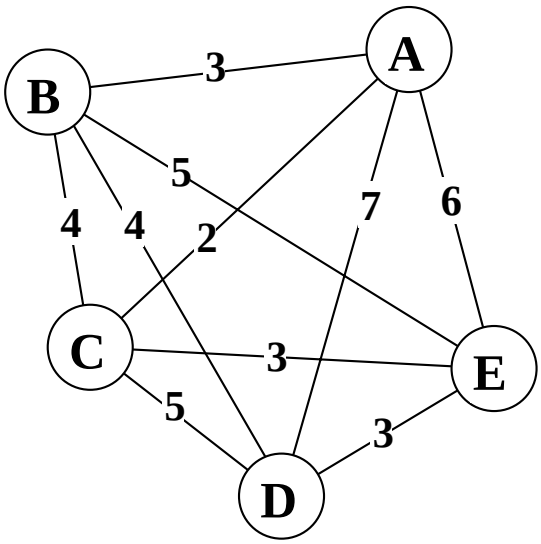

図 5.2　都市図

5.1　実装方法

本環境をベースに，TSP 用に都市マップ生成関数と適応度関数を作成し実装を行った．

5.1.1　染色体の表現方法

染色体の符号化には順列符号化を用いる．例えば，解集団中に $s = [1\ 3\ 4\ 2\ 5]$ で表される染色体 s がある場合，$1 \rightarrow 3 \rightarrow 4 \rightarrow 2 \rightarrow 5 \rightarrow 1$ という経路情報 (解) を意味する．なお，この経路 s の経路長 $T(s)$ は次式で与えられる．

$$T(s) = d_{13} + d_{34} + d_{42} + d_{25} + d_{51}$$

5.1.2　適応度

GA は本来，適応度を最大化するように動作する．しかし，TSP は距離の合計を最小化することが目的である．そこで，経路情報 s から算出した経路長 $T(s)$ について以下の式を用いて大小関係を反転させ，s の適応度 $F(s)$ として採用した．

$$F(s) = \frac{1}{T(s) + 1}$$

5.2　実験

本環境を適用して，TSP を実装した場合の計算機実験を行った．本節で使用するパラメータを表 5.1 に示す．

表 5.1　パラメータ

個体数	N
未更新限度数	T_{limit}
交叉確率	P_x
突然変異確率	P_m
交叉手法	OX, PMX, CX, UOX

5.2.1　実験 1

親の選択方法であるルーレット盤戦略，トーナメント選択の 2 通りに対して，交叉方法の違いにより解の値がどの程度変化するのか，実験を行った．このとき，交叉確率 P_x，および突然変異確率 P_m を 0.1 から 0.9 まで 0.1 刻みに設定した．

この結果ルーレット盤戦略では，OX について，1-point-right は $P_x = 0.9, P_m = 0.4$，1-point-left は $P_x = 0.9, P_m = 0.7$，2-point-inside は $P_x = 0.7, P_m = 0.7$，2-point-outside は $P_x = 0.9, P_m = 0.6$ でそれぞれ最良解を得ることができた．PMX について，1-point-right は $P_x = 0.9, P_m = 0.5$，1-point-left は $P_x = 0.8, P_m = 0.5$，2-point-inside は $P_x = 0.9, P_m = 0.4$，2-point-outside は $P_x = 0.9, P_m = 0.6$ でそれぞれ最良解を得ることができた．CX については $P_x = 0.7, P_m = 0.1$ で最良解を得ることができた．UOX について $P_x = 0.8, P_m = 0.1$ で最良解を得ることができた．

トーナメント選択では，OX について，1-point-right は $P_x = 0.8, P_m = 0.9$，1-point-left は $P_x = 0.5, P_m = 0.9$，2-point-inside は $P_x = 0.8, P_m = 0.9$，2-point-outside は $P_x = 0.7, P_m = 0.9$ でそれぞれ最良解を得ることができた．PMX について，1-point-right は $P_x = 0.6, P_m = 0.9$，1-point-left は $P_x = 0.8, P_m = 0.9$，2-point-inside は $P_x = 0.5, P_m = 0.9$，2-point-outside は $P_x = 0.9, P_m = 0.2$ でそれぞれ最良解を得ることができた．CX については $P_x = 0.9, P_m = 0.1$ で最良解を得ることができた．UOX について $P_x = 0.7, P_m = 0.9$ で最良解を得ることができた．

5.2.2　実験 2

5.2.1 項の実験 1 によって，各交叉方法についてのベストな交叉確率と突然変異確率が見つかった．この確率を用いてルーレット盤戦略とトーナメント選択のそれぞれについて，$N = 50, T_{limit} = 500$ と $T_{limit} = 1000$ とし，乱数列を変えて各 10 回実験を行い，その最良解と平均値を求めた．表 5.3 にルーレット盤戦略，表 5.2 にトーナメント選択の結果を示す.

表 5.2　ルーレット盤戦略による実験結果

交叉手法		P_x	P_m	$N = 50$			
				$T_{limit} = 500$		$T_{limit} = 1000$	
				Best	Average	Best	Average
OX	1-point-right	0.9	0.4	635	710.30	635	710.30
	1-point-left	0.9	0.7	736	772.60	657	754.50
	2-point-inside	0.7	0.7	711	793.50	676	773.60
	2-point-outside	0.9	0.6	685	781.30	779	847.40
PMX	1-point-right	0.9	0.5	708	777.80	708	771.60
	1-point-left	0.8	0.5	715	793.50	715	787.30
	2-point-inside	0.9	0.4	689	806.30	689	806.30
	2-point-outside	0.9	0.6	453	683.80	453	655.30
CX		0.7	0.1	593	680.40	593	680.40
UOX		0.8	0.1	356	・527.20	・356	535.30

表 5.3　トーナメント選択による実験結果

交叉手法		P_x	P_m	$N = 50$			
				$T_{limit} = 500$		$T_{limit} = 1000$	
				Best	Average	Best	Average
OX	1-point-right	0.8	0.9	286	320.10	278	316.50
	1-point-left	0.5	0.9	169	187.60	・165	・181.80
	2-point-inside	0.8	0.9	176	196.30	176	196.10
	2-point-outside	0.7	0.9	302	341.10	302	340.00
PMX	1-point-right	0.6	0.9	286	329.70	267	325.00
	1-point-left	0.8	0.9	308	333.90	308	333.80
	2-point-inside	0.5	0.9	274	311.00	274	304.30
	2-point-outside	0.9	0.2	171	189.20	171	189.20
CX		0.9	0.1	282	320.30	282	319.60
UOX		0.7	0.9	281	311.80	279	305.60

5.3　考察

実験 2 よりトーナメント選択が，ルーレット盤戦略より優れた解を求めるのに適しているといえる．T_{limit} を 500 から 1000 に増やしても，最良解の値が変わらないのは，すでに T_{limit} が 500 の状態で収束しているからである．

第 6 章

結論

本研究では，順列符号化された染色体を扱う GA 開発環境を構築した．また本環境を組合せ最適化問題の 1 つである，TSP へ適用した場合について説明した．5 章から，本環境を適用して TSP を実装したところ，新たに作成した関数は適応度算出関数と都市情報読込み関数だけであった．さらに本論文では触れていないが，本環境を使用して集合 2 分割問題，部品配置問題へ適用した結果，共に新たに追加した関数は適応度関数とそれぞれの問題に必要なデータの読込み部分であった．以上のことから，本環境を適用することで，GA を用いたプログラム作成時間の大幅な短縮をはかることができたと考えられる．

今後の課題として，他の符号化方法にも対応した環境の構築，より汎用性の高い環境の構築，形質遺伝を意識した交叉関数の作成があげられる．本環境において使用できる全ての交叉関数は，致死遺伝子の抑制に重点をおいた交叉方法である．このためこれらの交叉方法は，子孫に継承すべき優良な形質を壊してしまい，適応度の改善が図れなくなり，結果として本環境の性能を悪くしてしまう恐れがある．そこで，染色体中の優良な形質をもつ部分集合を上手く子孫に継承する交叉方法，すなわち形質遺伝重視型交叉を作成することで，本環境の性能が向上できると考えられる．

謝辞

本論文は，著者が 2001 年 7 月から 2003 年 2 月までの高知工科大学工学部情報システム工学科在学中に，同学科坂本研究室において行った研究の成果を記したものである．

本研究に対して，直接御指導・御教示を賜わった坂本明雄教授に深く感謝致します．

また，本研究や輪講の全般を通して有益な御指導・アドバイスをして下さったり，修論に忙しい中，著者の論文に目を通してくれた大学院生の登伸一氏にたいへん感謝致します．同じく院生で卒論の時にお世話になった友池貴之氏，研究室配属当時は C 言語の基礎を御教授してくれましたが，最近めっきり姿を見かけなくなった折橋祐一氏，両氏にも深く感謝致します．

さらに，研究室活動において，よく何かしらの食べ物が机においてあるが，来年は院生になる河内友彦氏，気がつくと煙草を吸いにいってるけど，やることはやってる西村章氏，徹夜して研究室でがんばっているが，よくイスで寝ている河野兼祐氏，彼らのおかげで素晴らしい研究室ライフを送ることができたことを，感謝したい．特に西村，河野両氏には同じ喫煙者ということで，卒論の合間に一緒に煙草を吸いに行き，小一時間くだらない話題で疲れた体を癒してくれたことに尚一層の感謝を送りたい．

また，となりの研究室ということで何かとお世話になった福本研究室のみなさん，大変感謝しています．

そして，来年から卒業研究にとりくむ坂本研究室の 3 回生諸君，これから就職活動もあり忙しいかもしれませんが，がんばってください．

最後に，著者が本大学入学時から今まで過ごし易い環境を整えて頂いた情報システム工学科の諸先生方に感謝の意を表します．

参考文献

[1] 萩原 将文 著，ニューロ・ファジィ・遺伝的アルゴリズム，産業図書，1995.

[2] Sadiq M. Sait and Habib Youssef; *Iterative Computer Algorithms with Applications in Engineering.* 邦訳: 白石 洋一 訳, 組合せ最適化アルゴリズムの最新手法, 丸善株式会社，2002.

[3] 北野 宏明 編, 遺伝的アルゴリズム，産業図書，1993.

[4] X. Liu, A. Sakamoto, and T. Shimamoto，“Genetic channel router”，*IEICE Trans. Fundamentals*, vol.E77-A, no.3, pp.492-501, 1994.

付録 A

関数マニュアル

A.1　はじめに

本マニュアルは，本研究で作成された各関数の使用方法，アルゴリズムを記述したものである．

大まかに分類すると以下のようになる．

1. 初期解生成関数 (fst_set)
2. 両親選択関数 (roulette，tournament)
3. 交叉関数 (ox，pmx，cx，uox)
4. 突然変異関数 (two，rev，rand)
5. その他の関数

5. その他には 1〜4 までの各関数の補助関数，本プログラムにおいての変数名の意味や，構造体の説明が含まれる．

A.2　関数の説明

以下ではそれぞれの関数について説明を行う．具体的な運用方法については A.3プログラム中での使用方法の節で説明する．

A.2.1　初期解生成関数 (fst_set)

- gene **fst_set(int n，int l);

初期解生成関数は返却値に gene 型のダブルポインタを返し，第 1 引数 n に染色体数，第 2 引数 l に染色体長を必要とする．返却されるダブルポインタは染色体数×染色体長の 2 次元配列となる．なお，gene 型の詳細については A.2.5 その他の関数の項で紹介する．

A.2.2　両親選択関数 (roulette，tournament)

- Pare *roulette(int *estimation);

  ルーレット盤上に適応度を反映した扇上の面積をとり，ランダムに発生した値によって選択される親を決定する方法．すなわち適応度の高い個体には広い面積が与えられ，適応度の低い個体には与えられる面積はわずかとなり，高い適応度の個体の選択される可能性が高くなるということになる．

- Pare *tournament(int *estimation);

  解集合をランダムに並べ換え，その集合を 2 分割し，部分集合を形成する．それら 2 つの部分集合より最も適応度の高い個体をそれぞれ親として選択する．選択後，再度解集合を撹拌し，同じ操作を実行する．このような操作を解集合に含まれる染色体の数だけ実行する．ルーレット盤戦略では適応度が 1 番低い解も選択される可能性が存在するが，トーナメント選択においてはその可能性はない．

上記 2 つの関数は共に，返却値に構造体 Pare 型のポインタを返し，引数には評価値の配列を必要とするため int 型のポインタで実装している．構造体 Pare の説明は A.2.5 その他の関数の項で説明する．

A.2.3　交叉関数 (ox，pmx，cx，uox)

- gene *ox(Pare *pare，gene **set，int k1，int k2);

- gene *pmx(Pare *pare，gene **set，int k1，int k2);
- gene *cx(Pare *pare，gene **set);
- gene *uox(Pare *pare，gene **set);

ox と pmx の関数は，返却値に交叉後の染色体を返すため，int 型のポインタを返す．第 1 引数 pare には親情報を保持しているリストを必要とするため Pare 型のポインタが必要である．第 2 引数 set には解集団を必要とするため gene 型のダブルポインタが必要である．第 3，4 引数にはそれぞれ切断点 k1，k2 が必要である．

cx と uox の関数は返却値に関しては，前述した ox と pmx の両関数と同じ型であるが，引数には第 3，4 引数の切断点は必要としない．なぜなら，これらの交叉方法は切断点を必要としないからである．

A.2.4　突然変異関数 (two，rev，rand)

- gene *two_mutate(gene *chromosome);

  ランダムに選ばれた 2 点の入れ替え操作を行う突然変異．
- gene *rev_mutate(gene *chromosome);

  ランダムに選ばれた 2 点間を逆順に並べ換える突然変異．
- gene *rand_mutate(gene *chromosome);

  ランダムに選ばれた 2 点間をランダムに撹拌する突然変異．

上記 3 つの関数は共に，返却値に操作後の染色体を返すため，gene 型のポインタを返す．引数には染色体を必要とするため，gene 型のポインタが必要である．

A.2.5　その他の関数

- Pare *choice_pare(int *estimation，char option);

  本関数は両親選択関数を内包しており，メイン関数においてどの選択方法を用いるか示すオプション文字を第 2 引数に渡すことで，ルーレット，あるいはトーナメントのどち

らかの選択方法を用いて親を決定する．返却値は両親選択関数の返却値をそのままメイン関数へと返却する．第 1 引数には評価値の配列が必要である．

- gene **crossover(Pare *pare，gene **set，char *xo_name);

  本関数は交叉関数を内包しており，メイン関数においてどの交叉方法を用いるかを示す文字列を第 3 引数に渡すことで，各交叉方法を決定する．返却値は 1 世代の子孫を生成し終わった時点での解集団を形成し返却する．すなわち各交叉関数において返却される染色体を，本関数において解集合として連結する．第 1 引数は両親のペアをあらわす構造体 Pare 型へのポインタであり，第 2 引数は親世代の解集団である．

- gene *mutation(gene *chromosome，char *mutation_name);

  本関数は突然変異関数を内包しており，メイン関数においてどの突然変異を用いるかを示す文字列を第 2 引数に渡すことで，どの突然変異を用いるか決定する．返却値は突然変異関数の返却値をそのままメイン関数へと返却する．第 1 引数には対象となる染色体を必要とする．

- void initialize(int argc，char **argv);

  本関数は環境の各関数で使用される変数を初期化する関数である．引数にはコマンドラインパラメータを必要とする．

- int *AllocInt(int size，char *name);

  本関数は第 1 引数で与えられたサイズだけメモリ領域を確保する関数である．領域確保に失敗した場合には第 2 引数で与えられる領域名を使ってエラーメッセージを出力する．

- int *rand_str(int size，int *string);

  本関数は第 2 引数で与えられた文字列をランダムに撹拌するものである．

- Pare *addNewData(Pare *set，int p1，int p2);

  本関数は構造体 Pare の各メンバに値を代入する関数である．

- int make_index(double *rate，double d_rand);

  本関数はルーレット関数内部において，親の番号を決定する際に用いられる関数である．

- int check_exist(int target，int end，int *string);

  本関数は第 3 引数で与えられる文字列中に，第 1 引数で与えられる数字があるかどうかを判定する関数である．返却値は該当しなければ-1 を返却し，該当するならば，何番目に存在するのかを判断し，その番号を返却する．

- void swap(int *string，int i，int j);

  本関数は第 1 引数で与えられる文字列中の 2 文字を入れ替える関数である．

- void bubble_sort(int *string，int end);

  本関数は第 1 引数で与えられた文字列を昇順ソートする関数である．

- 構造体 Pare

  構造体 Pare は P1,P2 に親となる染色体の番号 (染色体の番号は解集合において決定される) を格納し，*next には次の構造体へのポインタを格納する．すなわち構造体 Pare はリスト構造である．以下のプログラムソースは実際のプログラム中での定義，図 A.1 はそのイメージ図である．

```
typedef struct pare
{
   int p1;
   int p2;
   struct pare *next;
}Pare;
```

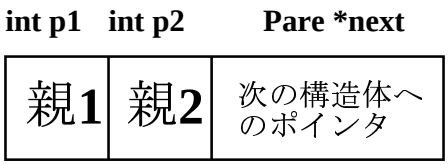

図 A.1　構造体 Pare のイメージ

- gene 型

  gene 型は int 型を typedef を使用して別の名称を付与させたものである．実際の動作については int 型と同等である．プログラム中での意味を判別しやすくするために gene 型を定義した．

A.3　プログラム中での使用方法

本節では実際にどのようにプログラムを組めば本環境を利用できるのかを説明する．

A.3.1　初期解生成関数

```
gene **set;
set = fst_set(5, 7);
```

以上のようにプログラム中で記述すると染色体数が 5 個，遺伝子長が 7 の 5 × 7 の図 A.2 のような初期解集合が形成される．

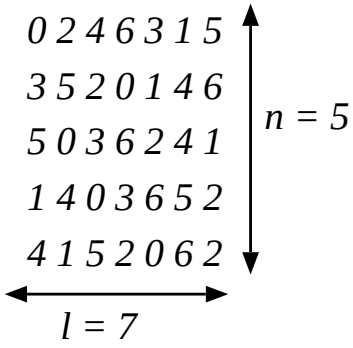

図 A.2　初期解集団

A.3.2　両親選択関数

```
Pare *pare;
pare = choice_pare(estivalue_set, 'r');
```

以上のようにプログラム中で記述するとルーレット戦略を用いて両親の選択を行う．なお，estivalue_set とは各染色体についての評価値を格納した配列である．この評価値の配列は利用者負担の元で生成される．第 2 引数は A.2 節の両親選択の項で述べた通り，ルーレット戦略:r とトーナメント選択:t の 2 通りある．

A.3.3 交叉関数

```
set = crossover(pare, set, "pmx_2_in");
```

以上のようにプログラム中で記述すると pmx-2-point-inside の交叉方法を用いて set の解集合から pare で与えられた親情報を元に交叉を実行する．なお，ここでの pare は既に両親選択後の値を持っており，set には解集団が格納されているものとする．交叉方法においては A.2 節の交叉関数の項にて説明した通り，ox，pmx，cx，uox の 4 種類があり，さらに ox，pmx においては ox_1_L や ox_2_out のように細分化されてある．

A.3.4 突然変異関数

```
*(set + 3) = mutation(*(set + 3), "rev");
```

以上のようにプログラム中で記述すると rev という突然変異方法を用いて解集合 set の 4 番目染色体に突然変異を起こす．突然変異の種類には rev の他に two と rand の計 3 種類用意してある．

A.3.5 その他の関数

ここでは上記以外の関数をどのようにプログラム中で組めばいいのかを説明する．

- void initialize(int argc，char **argv);

```
initialize(argc, **argv);
```

本関数については原則的にこの書き方のみである．なぜならば，本関数は GA 全体において使用される重要なパラメータを変数として取り込む関数だからである．

- int *AllocInt(int size，char *name);

```
int *string;
int size = 10;
string = AllocInt(size, "string");
```

以上のようにプログラム中で記述すると size 分領域が確保され，ポインタ string にその先頭アドレスが格納される．また，領域確保に失敗した場合，エラーメッセージが出力される．

- int *rand_str(int size，int *string);

```
string = rand_str(size, string);
```

以上のようにプログラム中で記述すると第 2 引数で得られた配列について第 1 引数の size 分だけランダムに撹拌した配列が得られる．

A.4　コマンドラインパラメータ

本節では，各種コマンドラインパラメータが表す意味，使用例について説明する．なお，各項目中でのデフォルト値は，コマンドラインパラメータの設定が無い場合に採択される設定値である．

- -t

  世代未更新限度数．デフォルト値は 50 世代である．

  使用例 ./hogehoge -t200

  200 世代の間最良解が更新されなければ終了する．

- -n

  解集合中の染色体数．デフォルト値は 10 である．

  使用例 ./hogehoge -n20

  染色体の数を 20 に設定する．

- -l

  染色体長数．デフォルト値では 10 である．

  使用例 ./hogehoge -l40

  染色体長を 40 に設定する．

- -m

  突然変異確率．デフォルト値は 0.05 である．

  使用例 ./hogehoge -m2.31

  突然変異発生確率を 231%に設定する．

- -x

  交叉採択確率．デフォルト値は 1.0 である．

  使用例 ./hogehoge -x0.473

  交叉採択確率を 47.3%に設定する

- -s

  疑似乱数列の種の決定．デフォルト値は 1 である．

  使用例 ./hogehoge -s3

  疑似乱数列の種を 3 に設定する．

- -c:r，-c:t

  両親選択方法の決定．デフォルトではルーレット戦略である．

  使用例 ./hogehoge -c:t

  両親選択方法をトーナメント選択に設定する．

- -m:two，-m:rev，-m:rand

  突然変異の決定．デフォルトではランダムに選ばれた 2 点の入れ替えである．

使用例 ./hogehoge -m:rev

突然変異をランダムに選ばれた 2 点間を逆順に設定する.

- 交叉方法の決定

  交叉方法の決定. デフォルトでは pmx_1_L である.

  使用例 ./hogehoge -ox_2_in

  交叉方法を ox-2-point-inside に設定する. なお, 以下に全ての交叉種類を載せる.

  pmx_1_L, pmx_1_R, pmx_2_in, pmx_2_out, ox_1_L, ox_1_R, ox_2_in, ox_2_out, cx, uox