平成 12 年度

学士学位論文

遺伝的アルゴリズムを用いた
複数経路探索法

Exploration Method of Various Routes
with Genetic Algorithm

1010369　　井上 祐介

指導教員　　坂本 明雄

2001 年 2 月 5 日

高知工科大学 情報システム工学科

要 旨

遺伝的アルゴリズムを用いた
複数経路探索法

井上 祐介

経路探索法の応用の一つとして、自動車用ナビゲーションシステムにおける最短経路決定がある。本論文では，遺伝的アルゴリズム（ Genetic Algorithm: GA ）を用いて指定したマップ情報の経路を探索する手法を提案する。本手法は，GA の多点探索という特徴を生かして，経路探索をする際に最短経路を選択するだけでなく，第 2，第 3 に短い経路の決定が可能である。このような経路は，従来最短経路問題で用いられてきた Dijkstra 法では容易に選び出すことはできない。更に本手法は，異なる重みを付けた複数の評価関数を導入することにより，経路長が短く，かつ経由するノードが類似しない複数の経路を同時に決定することが可能である。また，実際の地図のデータに適応した実験を行うことにより，本手法の有効性を確認した。

キーワード　遺伝的アルゴリズム，最短経路探索，複数の経路，Dijkstra 法

Abstract

Exploration Method of Various Routes with Genetic Algorithm

Inoue Yusuke

There is the shortest route decision in the navigation system for the car as one of the applications of the route search method. In this paper I would like to propose the technique of exploring route of denotation map file with a Genetic Algorithm (GA). When the feature of multi point search of GA is made the best use of, and it searches for the route, this technique is able not only to search for the shortest route but also decide the 2nd and the 3rd a short route. Such a route cannot be easily chosen by the Dijkstra method which has been used about the shortest route problem so far.In addition, this technique can decide two or more routes to which the route length is short and the node passed is not similar at the same time by introducing two or more evaluation functions which put different weight.Moreover, the effectiveness of this technique was confirmed by doing the experiment which adjusted to the data of an actual map.

key words genetic algorithm, shortest route, two or more routes, Dijkstra method

目次

図目次

表目次

第 1 章

まえがき

最短経路問題は，重み付きグラフ上で与えられた 2 頂点を結ぶ道の中でそれに沿ったパスの重みの総和が最小のものを求める問題である。重みは 0 以上の実数値とする。すべての重みが 1 のとき，最短経路は 2 頂点を最も少ないノード数で結ぶ経路になる。これは交通網で都市から都市への道路の長さが与えられているときに，ある都市から都市への最短な経路問題を求める問題など，いろいろな問題をモデル化したものである。

最短経路問題に対する解法で最も基本的なアルゴリズムとして，全探索法である深さ優先探索（Depth First Search: DFS）法，幅優先探索（Breadth First Search: BFS）法 [1] がある。いずれもグラフの 1 点を開始点としてすべての点，すべての辺を訪問する総当たり探索である。

これらは安定して最短経路が得られる反面，すべての点やすべての辺を訪問するため，探索に時間がかかるという欠点をもつ。幅優先探索法の一種である Dijkstra 法は短い探索時間で最短経路を得ることができる。そのため，この方法を応用した多くの手法が提案されている。

Dijkstra 法 [2] [3] を用いた経路探索法は，最短経路を安定して得ることができ，しかも探索時間が短いという利点がある。しかし，この手法は最短経路のみを求める手法であるため，一部を回避した経路や最短経路とは別の短い経路，また複数の経由点を通る最短経路を求めることは困難である。さらに，その探索法から複数の経路候補を一度の探索処理で選び出すことは難しい。

遺伝的アルゴリズム（Genetic Algorithm: GA）[4] [5] は，個体集団が世代交代を繰り返し，環境に適応した個体集団へと進化し最適解を求めていくアルゴリズムであり，あらゆる

最適化・探索の問題に適用可能な探索手法である。GA は，短い時間で実用的な解を得ることができること，複雑な組み合わせ問題を効率的に解くことができること，幅広い問題に対応できること等の長所をもつ。

実際に，ある頂点から他の頂点への最短路を見つける問題に還元される問題は多い。例えば，道路の経路探索（ナビゲーション）や，製品プロジェクトのスケジューリング問題など，さまざまな最短経路問題がある。

本論文では，GA を用いて最短経路のみならず複数の経路候補を示すことができる経路探索法 [6] を提案する。この手法を用いると，最短経路の次に短い経路探索や，渋滞や通行止めなどのように道路事情によって利用が望ましくない道を回避した最短経路を得ることができる。

また，距離が短い複数の解候補を選択すると，得られた解候補のもつ経路が酷似してしまうという問題が発生する。そこで本手法は，異なる重みを付けた複数の評価関数を導入することにより，酷似しない解候補を得られることができ，この問題を解決した。

本論文の構成は以下のとおりである。次の章では，従来法について説明する。3 章では遺伝的アルゴリズムの流れについて説明する。4 章では，本手法の GA の設定について紹介する。5 章では，Dijkstra 法と GA との比較を行う。6 章では，重み付きの経路探索を行った実験について述べる。最後の 7 章はまとめである。

第 2 章

従来法

点や辺を処理する方策あるいは順序には，様々な問題を解く大変有用で効率のよい方法をもたらすものがある。グラフのアルゴリズムで最も基本的なのは，すべての頂点や辺に何らかの処理を加えて探索することである。そのためには，それぞれの頂点や辺を 2 回以上調べることなく，しかも必ず 1 回は調べる組織的な手順が必要になる。

グラフを扱う場合の最も基本的なアルゴリズムとして，DFS と BFS の探索法がある。いずれも 1 点を開始点としてすべての点，すべての辺を訪問する。

DFS では，未訪問の隣接点があればそこへの前進操作を行い，その点で再び同様のことを行う。だだし，隣接点がすべての訪問ずみのときには，この点に到着した前進操作の出発側に一度戻り，再び別の隣接点へ前進操作を反復する。

BFS では，各点に接続する未訪問辺をすべて連続して訪問する。また，複数の解がある場合は，最も浅いゴールを最初に見つける。

DFS や BFS は，すべての点やすべての辺を訪問するため安定して最短経路を導き出すことができるが，その反面，計測する時間量がかかってしまうという欠点がある。そこで，短い時間で最短経路を探索する Dijkstra 法がある。以下に，この手法について示す。

Dijkstra 法とは，グラフ上の各節点への最短路を始点から一つずつ確定し，徐々に範囲を広げていき最終的にすべての節点への最短路を求めていくものである。具体的には，次のようなアルゴリズムとなる。

Step 1　出発点と連結されているノードとの間の距離を求める。（図 2.1（a））

Step 2　最小の値を持つノードに印をつけて確定する。（図 2.1（b））

Step 3　印をつけたノードと連結されているノードとの間の距離を求める。

Step 4　この時点で計算されているノードの距離の中で最小の値を持つノードに印をつけて確定する。(図 2.1 (c))

Step 5　これを全てのノードに印がつくまで繰り返す。

Step 6　最終的に各ノードに対して得られた値が，出発点からの最短距離になる。

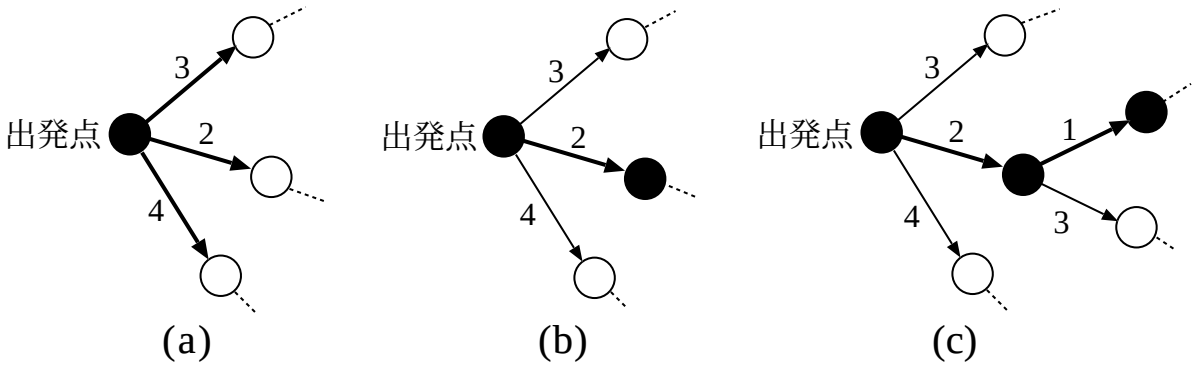


図 2.1　Dijkstra アルゴリズムの流れ

Dijkstra のアルゴリズムで最短経路が正しく求められる理由を以下に示す図 2.2 を用いて説明する。集合 V に含まれる頂点が訪問ずみの頂点，U が未訪問の頂点に対応する。

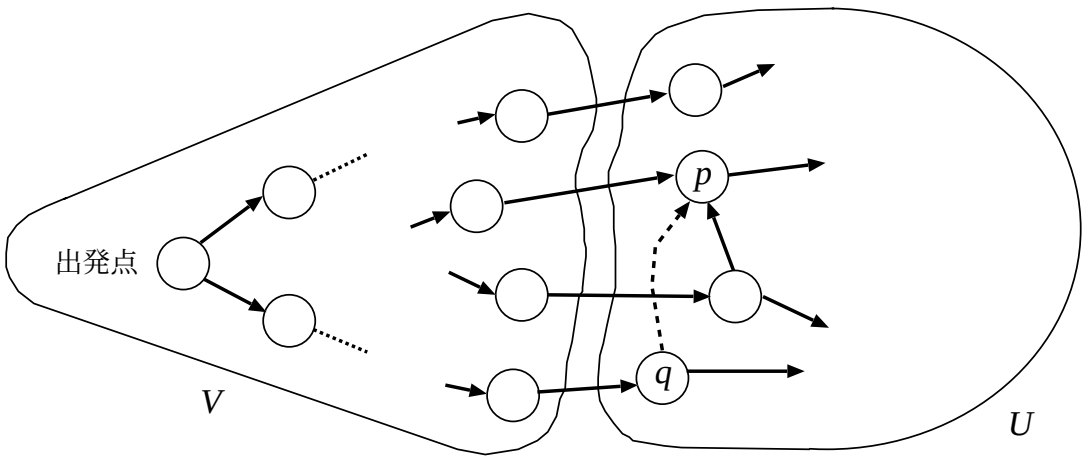


図 2.2　Dijkstra のアルゴリズムの正しさの証明

まず，U 内の各頂点のパスには，V に属する頂点だけを経由してその頂点に達する最短の道の長さが常に記録されている。これは，アルゴリズムの作り方から容易に確認できる。し

たがって，集合 U の中でパスの値が最小の頂点 p を選んだときに，p までの最短路が V に属する頂点しか経由しないことを証明することにより，この時点で p までの最短路の長さを確定できる。

p までの最短路が V に属する頂点だけを経由することは次のように証明できる。V に属しない頂点を経由して p に至る道があり，V から直接 p に達する道より短いものとする。これは，図 2.2 の q から p に向かう破線で示している。この道の上で V の外にある最初の頂点を q とする。このとき，q から p までの道の長さは正または 0 であるから，出発点から V に属する頂点だけを通って q に至る道の長さは，同じく p に至る道の長さより短くなければならない。しかし，これはありえない。この長さが最短のものを選んで p としたのだから，p に至る道は q に至る道よりも短いまたは等しいはずである。したがって，q を経由して p に至る道の方が短いという仮説は誤りである。

以上の証明からわかるように，Dijkstra のアルゴリズムが正しく動くためには，辺の重みが負でないことが本質的である。

この探索法は，短い時間で最短経路を選択できるという長所がある。しかし，複数の経路候補を一度の探索で選び出すことは難しい。さらに，類似しない解を発見するにはランダムに重みを付け，適切な解が得られるまで処理を繰り返さなければならない。

第 3 章

遺伝的アルゴリズム

遺伝的アルゴリズムは，生物の遺伝機構とダーウィンの進化論にアイデアを得て John Holland により発明されたアルゴリズムであり，適当なデータ構造をもつデータを生物の個体と見立てて，生物の進化を計算機上でシミュレートするアルゴリズムである。

まず，基本的な用語について説明する。GA とは，遺伝情報を伝える実態として染色体の集団が存在する。染色体の各位置にどのような遺伝情報が記述されるかが決まっているが，このような位置を遺伝子座という。この，遺伝子座に対してその形質を決定するコードを遺伝子という。そして，遺伝子の組み合わせのパターンを遺伝子型という。また，その遺伝子によって定義される形質を表現型という。

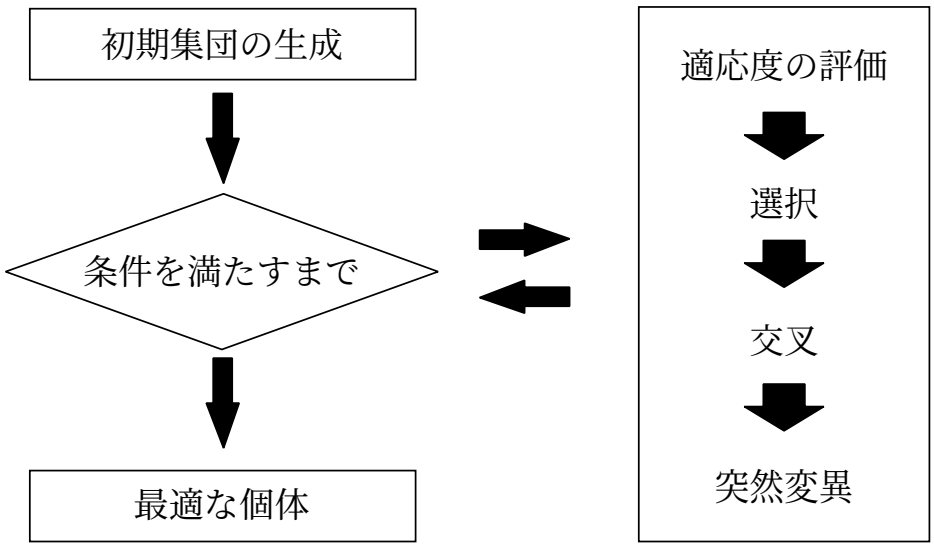


図 3.1　遺伝的アルゴリズムの処理手順

GA の処理手順を図 3.1 に示し，その流れを簡単に説明する。まず，遺伝子型を生成し，

初期集団の生成を行う。一般には，決められた個対数の染色体をランダムに生成する。次に各々の個体に対して適応度の評価を行う。最適解はその環境のもとでは最も適応度が高く，したがって高い適応度はより良い個体であるという評価をされるという適応度関数を用いる。各々の個体の適応度が決定されたら，それを基に選択淘汰を行う。基本的に，集団の中から生殖のための染色体を選び出す作業であり，適合する染色体は適合しない染色体よりも多くの子孫を平均して生み出す機構となる。次に，選択された個体を親として，遺伝子型の一部を入れ換える交叉を行う。最後に突然変異を加える。これは，ある確率で染色体の一部の値を変える操作である。このような世代交代を繰り返すことにより，最終的に最適解を示す個体を探索することができる。

GA を用いる場合には，あらかじめいくつかのパラメータを決定する必要がある。例えば，集団数，交叉確率，突然変異確率などであり，アルゴリズムの善し悪しはこれらの値に大きく依存する。これらのパラメータを適切に選ばなければ，最適解を求めるために多くの計算時間がかかったり，最適解を求めるどころか局所解に陥って最適解が得られないなどの問題が起こる。

本手法において，それらのパラメータがどのように設定されているか次章に示す。

第 4 章

GA のパラメータ調整

4.1　初期集団の発生

目的とする解を得るために解候補の集合内を探索するという考え方は，コンピュータサイエンスにおいては一般的であり，探索空間における探索と呼ばれている。本問題の探索空間は，マップ情報を基に指定した出発点と到着点で完結する経路集合である。まずマップ情報から隣接リストを生成し，そのリストを基に経路を表す表現型を生み出す。そして，得られた表現型を遺伝子型にコード化し，初期集団を発生させる。

4.1.1　隣接リスト

グラフの各点に対して，それに隣接する点の集合またはそれに接続する辺の集合を 1 本の線状リストとして格納することによりグラフの構造を表現できる。隣接リストは，このような点数に等しい本数の線状リストをひとまとめにしたものである。構造体を利用した多重グラフの隣接リストを図 4.1 に示す。

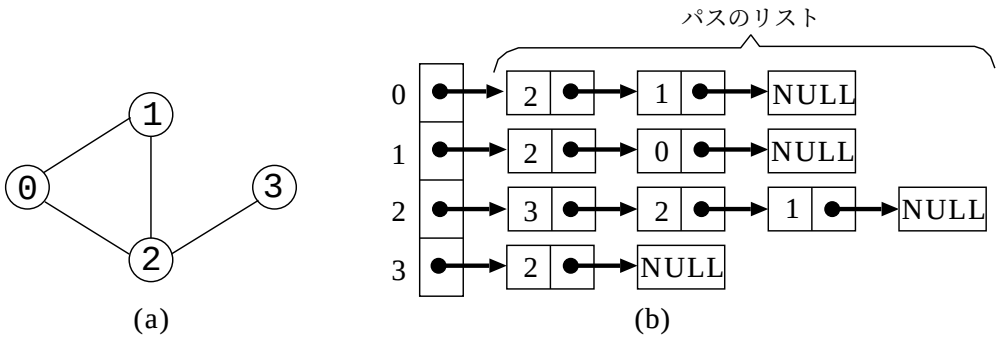


図 4.1　隣接リストの一例 (a) 多重無向グラフ (b) 頂点に関するデータ構造

図 4.1 の（b）は，（a）の多重無向グラフを基に構造体を利用したリストである。出発点であるノード 0 にはノード 1 と 2 が隣接しているため，1 と 2 の構造体が用意されそれぞれの場所に格納される。それ以外は隣接してないため，次の構造体には必然的に NULL が格納され，ノード 0 に関するデータ構造は終了する。以下同様に処理を行うことで，与えられた情報から隣接リストを生成する。

4.1.2　表現型の生成

単に遺伝子型の先頭から順に各遺伝子をランダムに選択していく方法によると，完結しない経路を表す致死遺伝子となる可能性が高くなり，初期集団を発生することが困難となる。このため，次のような手法で表現型を発生させる。

まず，隣接リストを基にして，出発点から隣接しているノードのうち一つをランダムに選択する。次に，選ばれたノードから次に向かうことのできるノードのうち一つを選択する。これを到着点に至るまで繰り返す。図 4.2 に多重無向グラフから表現型の生成を行う例を示す。

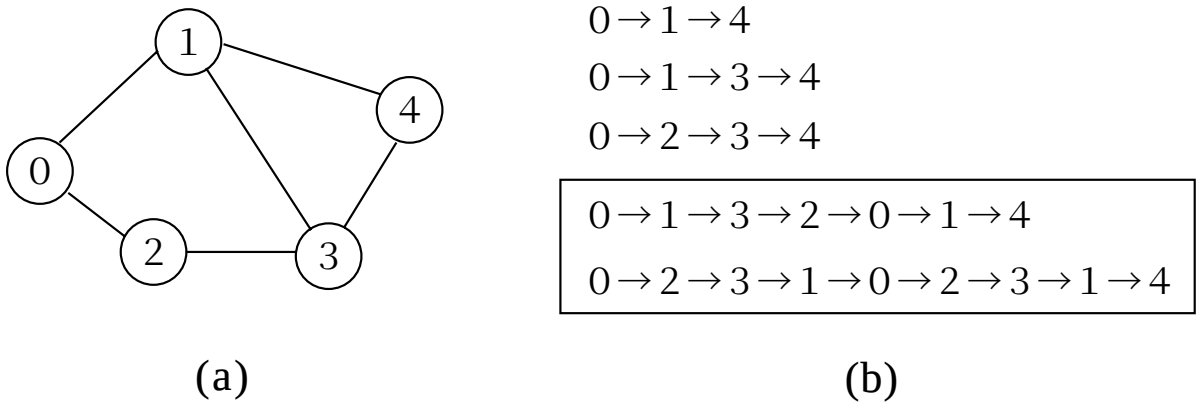


図 4.2　表現型の一例（a）多重無向グラフ（b）表現型候補

図 4.2 の（a）を，出発点を 0，到着点を 4 で完結する多重無向グラフとする。0 を出発し，それに隣接しているノードをランダムに選択する。そして到着点の 4 に着くまで繰り返される。その際，到着点に到着することなくループになる経路は致死遺伝子となる。

発生した表現型に図 4.2 の□で囲んであるような同一番号を複数回通過する経路が生成される場合がある。複数回通過することは，最短経路として除外され，また計算量の増加にもつながる。その通過を防ぐため，経路生成時に出発点からランダムに選択されたノードに印を付けていく。そして，経路探索中に印の付いているノードが再度選択された場合，その時点で致死遺伝子と認識し経路探索を終了する。その後，その致死遺伝子の情報は記憶されず，再度出発点から経路探索を行う。この処理により出発点と到着点で完結し，かつループをもたない経路を表す表現型を得ることができる。

4.1.3　遺伝子型の設定

本手法で用いる遺伝子型は，経由するノード番号を遺伝子とする。すべての遺伝子はその遺伝子座の番号のノードが接続しているノード番号のいずれかを表しており，実際に存在するパスの一つを情報としてもっているため，実際に存在しないパスを表現型にもつ個体は生成されない。しかし，経由するノードの情報のみをもっている遺伝子を経路探索問題に適用した場合，ランダムにノードを選択していくと，実際に存在しない経路を表す致死遺伝子を発生してしまう。

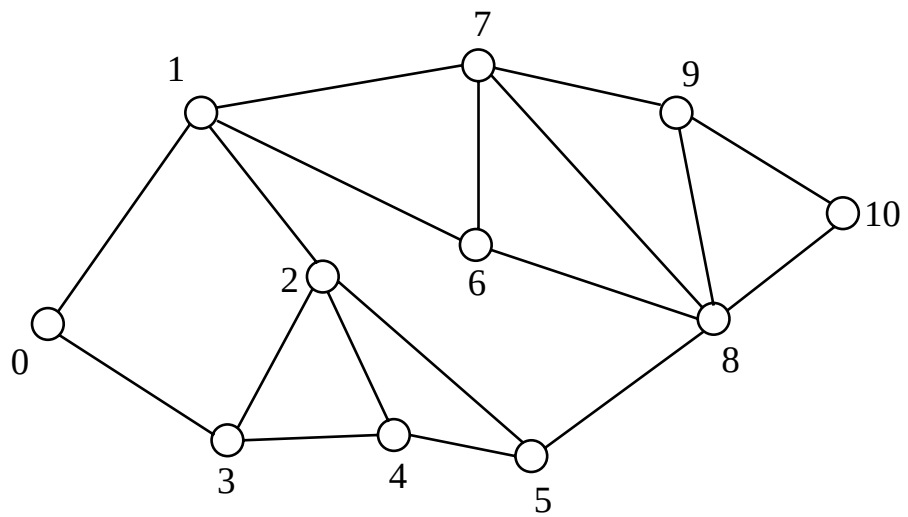
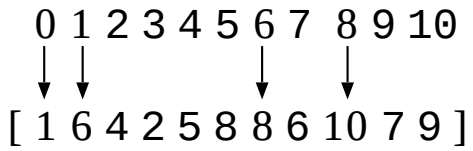


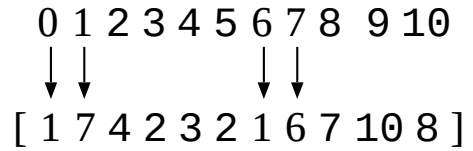
図 4.3　マップ情報

図 4.3 のマップ情報から 4.1.2 で示した発生方法で表現型を生成する。以下に出発点を 0, 到着点を 10 として 0 → 1 → 6 → 8 → 10 を表現型にもつ遺伝子型の生成過程を示す。



上段は遺伝子座の番号，下段は遺伝子型である。0 番目の遺伝子座に 1 が，1 番目の遺伝子座に 6 が，6 番目の遺伝子座に 8 が，8 番目の遺伝子座に 10 が格納されていることが確認できる。細字で示した遺伝子座にある遺伝子は，遺伝子座の番号のノードから次にいくことのできるノードの中からランダムに選ばれたノード番号を格納する。こうして，一つの遺伝子型が生成される。

例えば図 4.3 から致死遺伝子が発生したと考え，その遺伝子型の一例を以下に示す。



この遺伝子型は，0 を出発した後 1 → 7 → 6 → 1 を繰り返す表現型である。これは一つの経路情報としては記憶されず，致死遺伝子として認識される。このため，親となる個体，ならびに親から生成される子の個体は，ともに出発点から始まり到着点で完結する表現型を示す必要があり，初期個体もこの条件を満たしていなければならない。

4.2　評価関数の設定

本手法では評価関数（ Fitness ）を次式のように設定する。

$$Fitness = \frac{1}{\sum_{i}^{N} rlength(i)}$$

ただし，N を出発点から到着点までに経由するノードの個数，$rlength$（i）を i 番目と（i-1）番目のノード間の経路長とする。最短経路問題では経路長が短いほど適応度が高くなる評価関数を設定する必要があり，上記の式のように，経路長の総和の逆数により定義する。

4.3　選択淘汰

選択淘汰は，次項で述べる交叉によって，集団内から子孫を残すにふさわしい親となるかもしれない染色体を選ぶ操作である。選択は環境に対する各染色体の適応度に基づいて行われる。

今，集団内の染色体に対する適応度がすべて得られているものとする。このとき，まず次式から集団における各染色体の適応度の総和を求める。

$$F = \sum_{i=1}^{N} f_i$$

ここで，F は各染色体の適応度の総和で，f_i は各染色体の適応度関数値，N は集団内の染色体数である。この値が計算できれば，それぞれの染色体の選択確率 p_i は次式から計算できる。

$$p_i = \frac{f_i}{F} \quad (i = 1,2,\cdots,N)$$

次に，累積確率 q_i を計算する。累積確率 q_i は次式で計算できる。

$$q_i = \sum_{j=1}^{i} p_j = \frac{1}{F}\sum_{j=1}^{i} f_j \quad (i = 1,2,\cdots,N)$$

次の世代に残す染色体を決めるために，新たに乱数を発生させる。これを r で表す。今，累積確率 q_i と r との大小関係を順番に調べ，関係式

$$q_i < r < q_{i+1} \quad (i = 0,1,\cdots,N\text{-}1,\ q_0 = 0)$$

が満足されたとき，$i+1$ 番目の染色体を選択する。このプロセスを N 回繰り返すことによって，N 個の染色体が選択される。明らかに，$q_{i+1} - q_i$ が大きければ，乱数 r がその範

囲に入る可能性が高い。$q_{i+1} - q_i = p_{i+1}$ であるから，選択確率が高い染色体ほど選ばれやすく，逆に，選択確率の低い染色体ほど選ばれにくいということである。

それぞれの個体は，適応度に比例した大きさのルーレットの断片が割り当てられ，回転させたルーレットの矢印が停止した個体が次世代に対して親の候補に選ばれる。この選択を図に表すと図 4.4 のようになる。

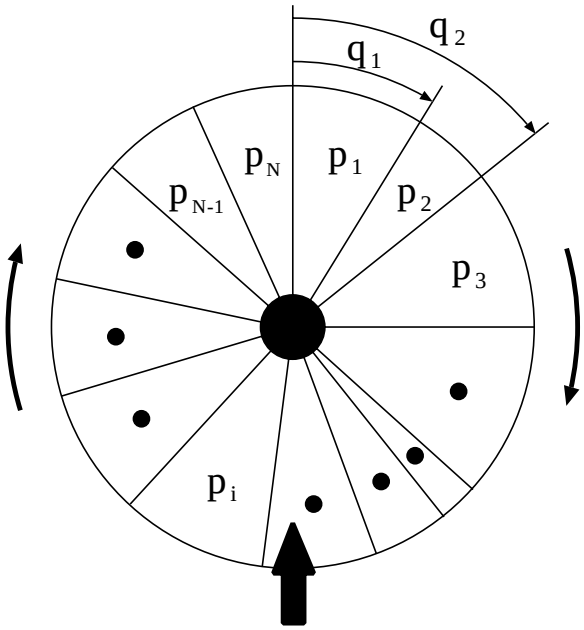


図 4.4　ルーレット戦略

選択は集団の中から個体を取捨選択するだけで，新しい個体を生成するわけではない。新しい個体の生成は交叉と突然変異によって行われる。本手法の選択淘汰は，集団から決められた数の個体を適応度に比例した確率で選択するというルーレット戦略を採用する。

4.4　交叉

交叉は，二つの個体間で染色体を組み換えることによって新しい個体を生成するもので，両親の優れた部分をうまく組み合わせて，子に継承させる操作である。この操作は，必ずしも良い個体を得られるとは限らず，中には適応度の低い個体や交叉手法によっては致死遺伝子が生成されることもある。本手法は経路探索を行うものであり，交叉の後に得られた個体

は，指定された出発点と到着点が完結する経路集合でなければならない。そこで次のような交叉を行う。

まず，初期集団からルーレット戦略によって選ばれた二つの個体より，一つ目の子を生成する。選ばれた二つの親個体から出発点のノード番号の遺伝子座にあるノード番号のうちからランダムに一つを選択して，一つ目の子の同じ遺伝子座に格納する。格納されたノード番号と同じ番号の二つの親の遺伝子座にあるノード番号のうちからランダムに一つを選択し，子の同じ遺伝子座に格納する。これを到着点に達するまで繰り返す。経由しなかったノード番号に関しては親の個体の遺伝子のどちらかをランダムに選択し格納する。二つ目の子に対しても同じ操作を行う。交叉の例を図 4.5 に示す。ただし，遺伝子座は [ ] で示し，経路はそれぞれの遺伝子座の右に示す。

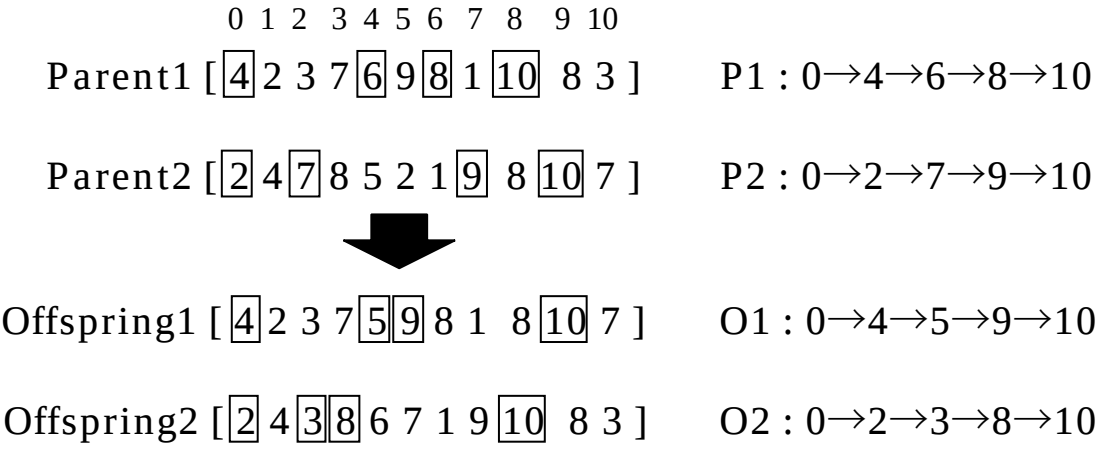


図 4.5　交叉方法

まず 0 番目の遺伝子座にある親 1 の 4 と親 2 の 2 の遺伝子からランダムにどちらかを選択し，子 1 の 0 番目の遺伝子座に格納する。親 1 の 4 によって，4 番目の遺伝子座にある親 1 の 6 と親 2 の 5 のうちどちらかをランダムで選択し，子 1 の 4 番目に格納する。これを到着点に達するまで繰り返す。経由しなかったノードについては，親の個体の遺伝子からランダムに選択する。子 2 も同じ操作で生成する。

本手法の交叉で行うと，最低 1 区間以上どちらかの親個体の経路を引き継ぎ，かつ実際に存在する経路を表す子の個体が生成される。これは，親個体で経路に関係しなかった部分の

遺伝子が交叉の結果，子の個体の表現型に発現するのもである。

場合によっては，親個体と同様の子の個体が生成される。この交叉方法によって生成される致死遺伝子は，個体の表す経路にループをもつものに限られ，選択した親の遺伝子型や交叉の操作手順により，生成された子の表現型における経路が無限ループとなる現象が起こる。この場合，出発点から再度交叉をやり直すことで，致死遺伝子でない子の個体が生成される。

上で述べたように致死遺伝子でない子を生成することは可能であるが，交叉処理を再度行うことは，直接計算時間の増加につながる。このような処理が必要とされる問題は，一つの遺伝子型中に同じ遺伝子が多く存在する個体が増殖した場合に，生成される子の個体が経路を表さない致死遺伝子となる確率が高くなることによって引き起こされる。ここで，図 4.5 で示した Parent1 と 2 の遺伝子型中に同じ遺伝子を多く存在させ致死遺伝子を生成したものを図 4.6 に示す。

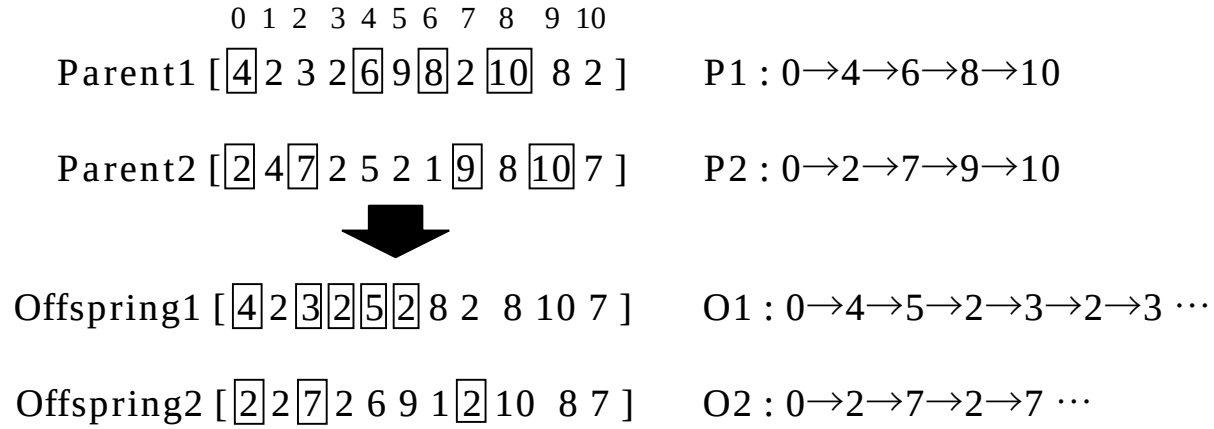


図 4.6　致死遺伝子

さきほど述べた交叉方法で Offspring を生成した結果，双方の子の表現型である経路が無限ループになっていることが確認できる。これは，ある一つの遺伝子座に格納される遺伝子が，その遺伝子座の番号のノードが接続しているノード番号のみであることから，ノード n が M 個のノードと接続しているマップを対象とした場合，遺伝子型中に同じ遺伝子 n が現れる遺伝子型の個数は最大で M であり，上の問題は M が遺伝子型の長さに比べて十分に

大きな場合にのみ生じる。したがって，多数のパスが入り込むノードをもつマップを対象とする場合には上の問題が生じる可能性があり，パラメータをうまく調節して致死遺伝子の増加を避ける必要がある。

4.5　突然変異

突然変異は，染色体上のある遺伝子を一定の突然変異確率で他の対立遺伝子に置き換えることにより，交叉だけでは生成できない子を生成して個体群の多様性を維持する働きである。図 4.7 は突然変異の一例である。

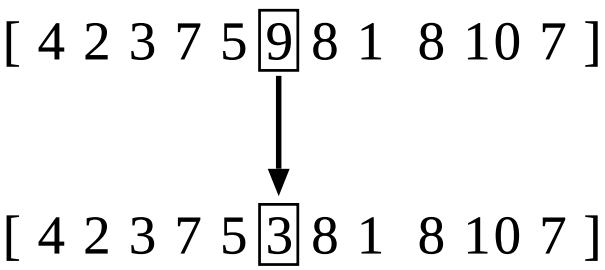


図 4.7　突然変異

図 4.7 の上段にある遺伝子座からランダムで一つ遺伝子を選択する。そして，選択された遺伝子の経由するノードから，ランダムで選択された遺伝子を下段の同じ遺伝子型に格納する。

第 5 章

比較と探索

本手法の特徴を示すとともに，最短経路探索，経路に重みを付けた最適経路探索の二通りについて実験を行った。経路に重みを付けて探索を行うことにより，そのパスを回避した最短経路を導き出す。

5.1 最短経路探索

ここでは，一つのマップ情報から出発点および到着点を始終点として最短経路を探索する。本章では，項知見の土佐山田駅－高知駅間のマップ情報に対して実験を行う。そのマップ情報に含まれる駅や交差点を示すノード数は 63，道路を示すパス数は 102 である。本実験において使用したパラメータを表 5.1 に示す。このパラメータは，実験を繰り返し行ったうえで最も良かった数値である。(付録 A 参照)

表 5.1　本章の実験による各パラメータ値

個体数	終了条件	交叉確率	突然変異確率
50	40 世代無進化	20 %	30 %

表 5.1 のパラメータを用いて実験を行った結果を図 5.1 に示す。これは，出発点を A の土佐山田駅，到着点を B の高知駅としての最短経路で，Dijkstra 法によって求めた経路と一致する。

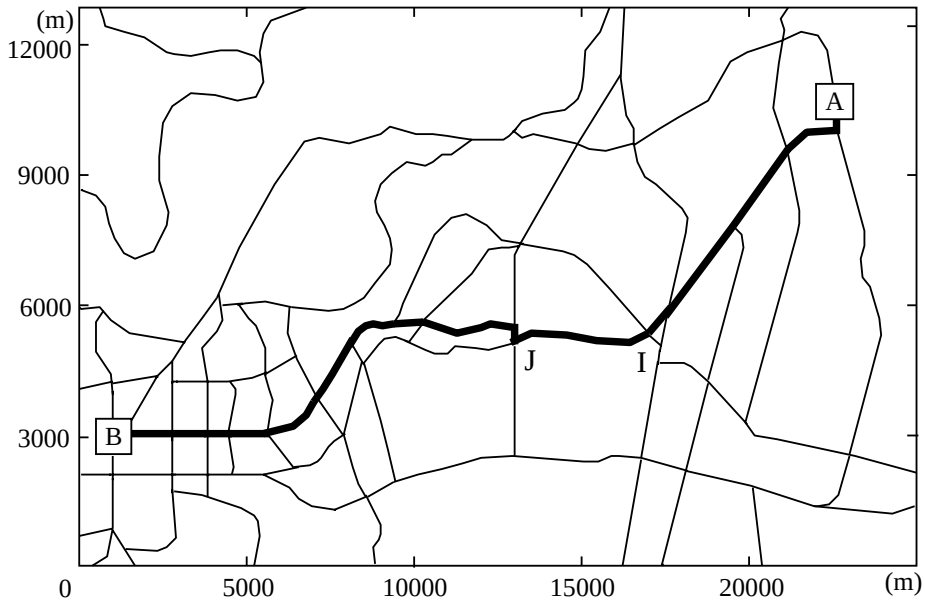


図 5.1　最短経路

表 5.2　経路探索の計算時間

本手法	0.0936s
Dijkstra 法	0.0096s

表 5.2 を見てわかるように，最短経路を求めるのは，Dijkstra 法の方が明らかに早いことが確認できる。しかし本手法では，初期集団の発生，選択淘汰及び交叉，突然変異の操作によって個体ごとにこれらの処理を行うことができ，並列処理が可能になると Dijkstra 法と比べても探索時間の上で問題はない。なお時間の計測に用いた CPU は，PentiumⅡ-400MHz である。

5.2　最短 2 番目経路探索

GA によって得られた 2 番目に短い経路を図 5.2 に示す。この経路探索は，Dijkstra 法では容易に導き出すことはできない。

5.2　最短 2 番目経路探索

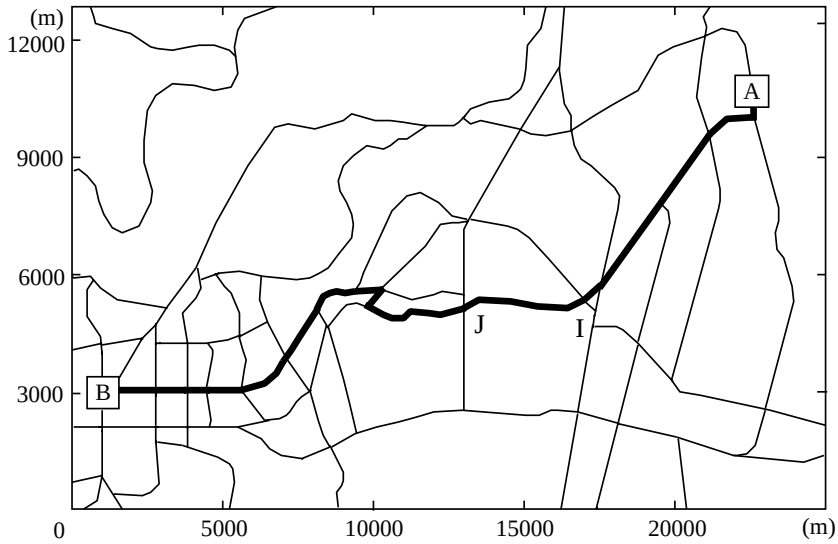


図 5.2　最短 2 番目経路

Dijkstra 法では，最短経路を求めるのが主であるため，図 5.2 に示す経路を導き出すには困難である。そのため，本手法は Dijstra 法に比べて有利であるといえる。

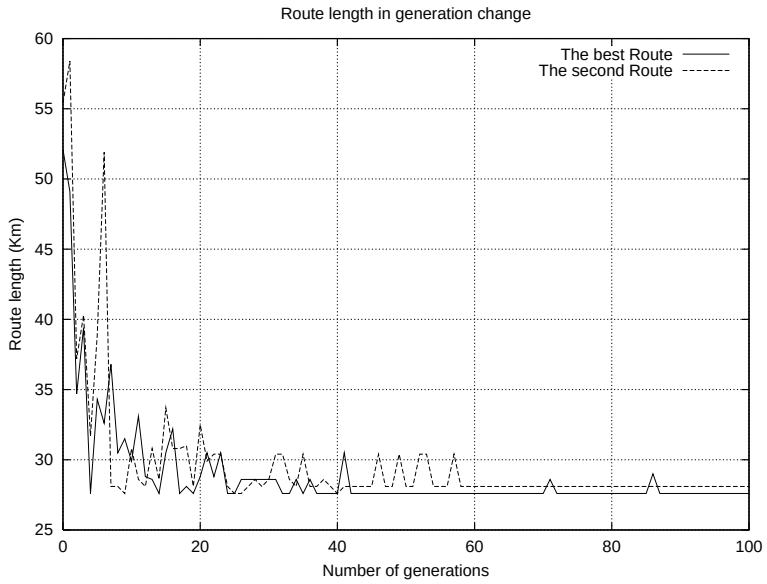


図 5.3　世代交代に伴う経路長の推移

本実験で設定した終了条件の妥当性を検証するために，世代交代に伴う個体の表す経路長の推移を調べる。図 5.3 は，高知駅と土佐山田駅間のマップ情報を用い，本章の実験で設定

した出発点と到着点で個体数を 50 とし，100 回の世代交代を行ったグラフである。実線は最短経路の経路長，破線は次に短い経路の経路長である。適応度は経路長の逆数と設定しているので，世代交代に伴い経路長が減少し収束していくことが確認できる。

5.3　重みを付けた最短経路探索

土砂で通行止めになったり，渋滞が起こったきなどの実際の道路情報の入手により，図 5.1 の経路 I–J の通行が好ましくない場合を考える。この区間は，他のノードより多く重みを付け，これを評価関数に用いて探索を行う。使用する各パラメータは表 5.1 と同様である。探索結果を図 5.4 に示す。

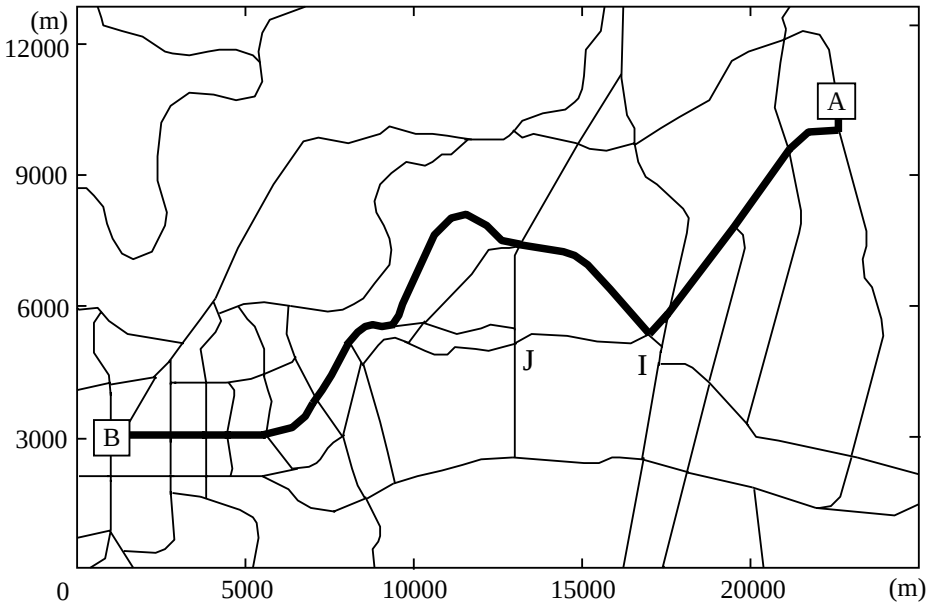


図 5.4　重み付き最短経路

図から確認できるように，出発点である A の土佐山田駅から到着点である B の高知駅に向かう経路で I–J 間を回避した最短経路である。これは，経路に重みを付けた上で Dijkstra 法によって求めた経路と一致する。

5.4　考察

アルゴリズムの終了条件を 40 世代無進化に設定しているが，この値を大きくするほど最短経路を発見する確率は上がる。しかしその反面，計測する時間が増すため，終了条件を大きくすることは望ましくない。（付録 A 参照）

Dijkstra 法による最短経路探索は，GA に比べて経路を短時間で導き出すことができる。しかし，最短経路とは異なる短い経路を導くには困難である。本手法では，Dijkstra 法より多少時間がかかるが，最短経路のみならず 2 番目 3 番目と短い経路を発見でき，なおかつ経路に重みを付加することでその経路を回避した最短経路を導き出すことが可能である。この点においては，Dijkstra 法より有効的と思われる。

第 6 章

複数経路探索

前章で示した最短経路や指定した道路を回避した経路を導きだすには Dijkstra 法を用いても可能であるが，ここでは GA の特徴を生かした複数経路候補の探索手法を示す。指定したマップ情報を領域分割して探索することで，酷似しない経路を導き出すことができる。

6.1 探索手法

前章での，図 5.1，5.2 の探索結果を比較すると両者の経路が酷似していることがわかる。本論文で対象としているカーナビゲーションシステムにおいて，最短経路にユーザが満足しない場合に示す次候補は，最短経路に酷似しない経路を探索する必要がある。そこでこの問題点を解消するために，図 5.4 で示した重み付き最短経路探索を応用した手法を示す。

Step 1 指定したマップ情報から探索範囲を複数の領域に分割する。

Step 2 それぞれの領域に異なる重みを付けた評価関数，およびどの領域にも重みを付けない評価関数を準備する。

Step 3 解候補を導き出す。

Step 4 ルーレット戦略で選ばれた個体から，高い評価関数を $\frac{\text{全個体数}}{\text{分割領域数}}$ 個 選択する。

Step 5 分割した領域数分 Step3 に戻り，処理を繰り返す。

Step 6 処理終了後，分割した領域を通る最良解の経路候補が選ばれる。

このように最短経路及びそれぞれの領域を通過する最短経路を得ることが可能となる。

最短経路として得られた経路探索と異なる経路が要求された場合に，Dijkstra 法ではパラメータを変更してはじめから探索をやり直さなければならず，探索をやり直した場合でもこのような経路が得られるとは限らない。しかし，GA の場合は探索の過程において多くの解候補を生成するため，これらの解候補を記憶させることで 1 回の探索処理により解を探索することが可能である。

6.2　領域指定した探索結果

あるマップ情報を基に探索範囲を 4 分割して重みを操作した後，前章と同様に指定した区間の探索を行う。

図 6.1　最短経路

実験は，土佐山田町の中心部のマップ情報で行う。図 6.1 に示す地図は，出発点を X の土佐山田駅，到着点を Y の高知工科大学とし，ノード数は 111，パス数は 174 である。ここで示されている経路は X-Y 間の最短経路で，点線は地図を分割したものである。実験で用いた各パラメータを表 6.1 に示す。（付録 B 参照）

表 6.1　本章の実験による各パラメータ値

個体数	終了条件	交叉確率	突然変異確率
150	60 世代無進化	90 %	70 %

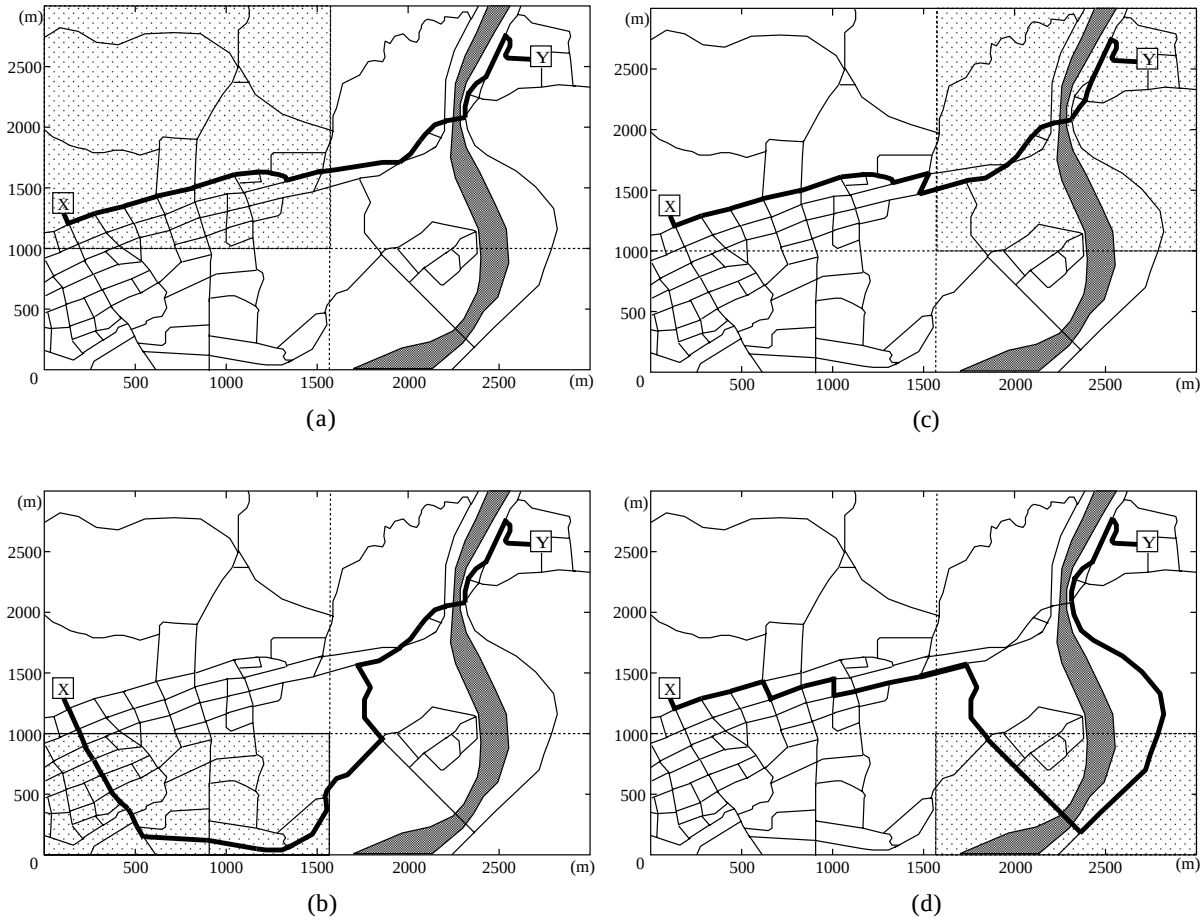


図 6.2　重み付き最短経路（a）左上中心（b）右上中心（c）左下中心（d）右下中心

分割した領域に重みを付けた探索結果を図 6.2（a）～（d）に示す。いずれの経路も各領域を通る最短経路であり，経由ノードが酷似していないことがわかる。

実験では四つの方形領域に分割したが，実際にはマップ情報から得られる地形などに基づいて分割することが予想され，本手法はこのような領域へも適用できる。以上の探索は，異なった評価関数を用いて独立に複数回の GA 処理を行うことによっても可能である。

6.3　考察

本問題では，用いたマップ情報やマップの領域分割の仕方，また領域への重みの付け方によっては，得られる複数の解が互いに一致，もしくは類似する可能性がある。しかし本手法によれば，探索過程で各々の分割領域ごとの解候補を互いに比較することにより，類似した解を回避することができる。Dijkstra 法の場合，分割領域数並列に実行したとしても，処理の過程で互いの解を比較することは不可能で，類似しない解を発見するためには，ランダムに重みを付けて適切な解が得られるまで処理を繰り返さなければならない。この点において本アルゴリズムは Dijkstra 法，並びに Dijkstra 法を複数回適用した場合と比較して有利である。

第 7 章

むすび

本論文では，最適解探索法の一つである GA を用いた経路探索法を示し，実験によりその有用性を確認した。最短経路では，Dijkstra 法と同様の探索結果を得ることが可能なだけでなく，探索過程で多数の解候補を生成する GA の特徴を利用し，最短経路以外の経路候補を 1 回の処理で探索することができた。

また，GA は評価関数の操作が容易であり，探索範囲を分割し，それぞれの領域に重みを付けることにより，距離が短く，酷似しない経路の探索が可能となった。

本手法で用いた GA の計算時間は，ノード数よりは繰り返しの世代数により大きな影響を受けるので，第 1 世代目の解空間が適切な大きさになるように調節すれば計算時間の増加はそれほど大きくはないはずである。つまり，探索ノード数の大きなグラフになればなるほど，GA が Dijkstra 法を上回ると考えられる。

今後の課題として，ユーザの目的地が多数ある場合の最短経路，および複数の経路候補を同時に決定する探索手法を考慮し検討していきたい。

謝辞

本論文は，著者が 1999 年 7 月から 2001 年 3 月までの高知工科大学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活動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，テニスのうまい情報システム工学科の坂本明雄教授に深く感謝致します。

また，本研究の全般を通して有益な御指導・御助言を賜わりましたギターのうまい大学院生の橋本学氏に深く深く感謝致します。

さらに，家庭をもちながら勉学に励む大学院生の久保真理子氏，カップラーメンを毎日のごとく食べ GA や第二種情報処理の勉強会のリーダーをした登伸一君，自分の趣味ばかりをやって研究を疎かにしているが言語に関してかなり強い神谷将司君，車のマフラーを交換し爆音で乗り回しているが文章力には強い有賀洋介君，最近姿を見ないが以前は毎日ネットを 3 つ立ち上げて同時にチャットをしていた折橋祐一君，現在彼女に無我夢中であまり研究室に顔を見せないが会ったときにはおもしろい発現をする横谷将樹君，途中にこの研究室に配属されたがほとんど研究室に姿を見せない山崎聖太郎君，唯一の女性で今は事情があって研究室から姿を消している山下由紀子さんに対し感謝の意を表します。

そして，坂本研究室の 3 回生である，亀本学君，木村大樹君，志摩浩君，一恩邦至君，栃木隆道君，これから行う研究を頑張って下さい。

参考文献

[1] 渡邊敏正，“データ構造と基本アルゴリズム”，共立出版，2000

[2] Brian W.Kernighan，Dennis M.Ritchie，“プログラミング言語 C”，共立出版， 1999.

[3] 石畑清，“アルゴリズムとデータ構造”，岩波書店，1989.

[4] Melanie Mitchell，伊庭斉志，“遺伝的アルゴリズムの方法”，東京電気大学，1997.

[5] 北野宏明，“遺伝的アルゴリズム”，産業図書，1993.

[6] 稲垣潤，長谷山美紀，北島秀夫，“遺伝的アルゴリズムを用いた経路探索における複数経路候補の決定法”，信学論 (D-I)，vol.J82-D-I，no.8，pp.1102-1111，Aug.1999.

付録 A

実験データ（最短経路）

第 5 章で最短経路を求める際の実験データを以下に示す。

個対数	終了世代数	交叉確率	突然変異確率
M	Tstop	Pc	Pm

- M = 50, T = 20 として各 10 回の試行における最良解（重みの総和）

Pc\Pm	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	393	693	693	626	570	715	626	635	636
0.2	693	673	693	620	590	639	542	680	664
0.3	648	659	618	673	393	693	673	582	693
0.4	635	639	693	393	393	693	416	620	601
0.5	555	695	693	597	693	795	464	673	709
0.6	540	693	393	494	416	420	693	468	704
0.7	693	680	639	589	・333	393	614	597	693
0.8	633	693	718	393	525	704	464	810	788
0.9	625	608	608	583	525	555	542	620	704

- M = 50, T = 30 として各 10 回の試行における最良解（重みの総和）

Pc\Pm	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	455	412	455	433	524	484	409	522	493
0.2	412	438	455	533	455	635	445	649	・286
0.3	445	455	462	452	455	315	423	505	462
0.4	460	455	455	513	455	452	423	420	462
0.5	412	462	466	455	501	431	445	455	465
0.6	445	462	446	635	315	452	473	455	315
0.7	484	462	462	513	462	474	455	583	445
0.8	455	455	446	494	516	455	402	454	513
0.9	455	412	430	467	445	640	440	525	315

- M = 50, T = 40 として各 10 回の試行における最良解（重みの総和）

Pc\Pm	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	319	319	319	319	319	315	305	319	655
0.2	490	319	・276	319	290	315	369	305	320
0.3	319	394	476	305	447	315	319	276	486
0.4	319	422	315	319	456	490	319	319	371
0.5	319	319	319	315	319	319	319	451	404
0.6	319	342	305	319	487	387	286	319	319
0.7	319	328	319	342	313	305	490	319	305
0.8	305	319	410	319	319	319	319	319	319
0.9	319	315	305	315	305	319	319	490	367

- M = 50, T = 50 として各 10 回の試行における最良解（重みの総和）

Pc\Pm	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.1	319	319	305	290	319	315	305	368	655
0.2	490	319	・276	319	315	315	369	319	320
0.3	319	394	476	319	447	315	319	・276	457
0.4	319	422	422	319	490	456	490	319	371
0.5	319	319	315	319	290	451	319	453	319
0.6	338	305	319	487	448	286	319	319	319
0.7	319	328	315	319	342	313	319	338	319
0.8	305	319	410	319	319	319	319	319	319
0.9	305	490	305	315	・276	319	319	490	367

— 探索結果で最良解を得た Cost と Time —

		総費用	Time (s)
M［50］, T［40］	Pc［0.2］, Pm［0.3］	19616	0.0936
M［50］, T［50］	Pc［0.2］, Pm［0.3］	21696	0.1337
	Pc［0.3］, Pm［0.8］	38800	0.19603
	Pc［0.9］, Pm［0.5］	116864	0.41998

付録 B

実験データ（複数経路）

第 6 章で最短経路を求める際の実験データを以下に示す。

- M = 150, T = 60 として各 10 回の試行における最良解（重みの総和）

Pc\Pm	0.1	0.2	0.3
0.1	247,308,421,300,357	247,252,525,259,689	244,243,806,490,458
0.2	246,395,552,453,947	247,229,638,608,437	258,280,634,547,389
0.3	237,235,524,491,571	247,251,447,451,415	259,232,541,690,604
0.4	248,245,534,599,782	240,232,597,547,454	252,259,576,408,435
0.5	240,234,790,574,644	248,222,818,550,395	244,233,268,815,644
0.6	241,262,262,647,406	265,253,365,408,517	375,224,345,478,789
0.7	404,271,394,294,793	263,242,557,502,854	358,233,423,429,425
0.8	358,367,399,436,369	363,276,362,258,411	374,244,531,298,438
0.9	436,243,427,236,416	358,255,324,383,387	406,223,278,357,384

Pc\Pm	0.4	0.5	0.6
0.1	261,243,578,742,499	243,255,258,439,391	255,234,644,603,665
0.2	245,261,464,429,430	238,222,384,570,518	272,244,429,484,763
0.3	230,251,377,357,360	238,223,637,770,328	243,246,394,523,549
0.4	243,266,423,639,368	242,226,621,647,366	267,222,379,483,468
0.5	238,246,343,397,424	259,267,311,782,360	269,240,475,433,367
0.6	370,249,744,361,433	242,234,269,383,379	239,357,308,242,388
0.7	396,226,432,385,370	273,237,522,371,428	295,233,467,358,414
0.8	371,235,378,255,366	239,260,268,362,402	237,252,300,236,369
0.9	260,235,331,409,414	365,248,256,240,443	242,234,381,236,359

Pc\Pm	0.7	0.8	0.9
0.1	258,258,562,398,422	243,251,442,637,378	233,229,354,367,361
0.2	235,249,445,818,798	249,238,273,307,415	251,231,311,236,448
0.3	236,234,517,820,665	254,267,356,432,352	241,237,256,390,358
0.4	240,240,380,335,399	229,235,441,255,286	235,222,331,382,370
0.5	237,374,559,363,372	246,228,282,233,439	251,225,402,228,427
0.6	363,222,359,251,360	242,243,547,313,395	240,276,454,311,365
0.7	240,252,539,363,373	256,244,297,334,368	292,279,288,242,361
0.8	358,258,347,428,356	232,263,467,266,365	256,247,256,335,362
0.9	• 222,222,256,236,352	352,256,425,236,421	355,359,450,236,383