

巡回セールスマン問題を解く枝組み立て交叉 (Edge Assembly Crossover) の拡張遺伝子交叉オペレータ交代法による性能改善

高 橋 良 英*

A Performance Improvement of Edge Assembly Crossover to Solve the Traveling Salesman Problem through Extended Changing Crossover Operators

Ryouei TAKAHASHI*

Abstract

In order to improve performance of solving the traveling salesman problem (TSP), we evaluate functionality and performance of Extended Crossover Operators (ECXO) which is an improvement of both of Genetic Algorithms (GAs), Ant Colony Optimization (ACO) and Simulated Annealing (SA). ECXO can substitute the current crossover operator for another suitable crossover operator at any time. In our study ECXO uses EX (or ACO) and EXX in early generations to create local optimum sub-paths and it uses EAX (Edge Assembly Crossover) to create a global optimum solution after generations. Our C experiments using the 198 city problem d198 in TSPLIB show that ECXO as well as EAX can provide us with the same optimum solution whose cyclic path's length is 15,780 and that ECXO can improve the performance of EAX in point of computer execution time.

Keywords: TSP, ECXO, GAs, ACO, EAX, SA

1. はじめに

巡回セールスマン問題 TSP (Traveling Salesman Problem) [2] を遺伝的アルゴリズム (GA: Genetic Algorithm) [1] により解く新しい手法として EAX (Edge Assembly Crossover) [9] が着目されている。EAX の有効性は国内外の論文で検証されている [4]。EAX は両親の枝を相互に交換して局所的に最適な部分巡回回路を構成した後、その部分巡回回路を結合して大域的に最適な解を構成する方法である。EAX

は従来の枝交換交叉 EXX (Edge Exchange Crossover) [6] やサブツアー交換交叉 SXX (Sub-tour Exchange Crossover) [5] の機能の拡張である。本研究では、拡張遺伝子交叉オペレータ交代法 (ECXO: Extended Changing Crossover Operators) [16] の検討の一貫として主制御対象オペレータを EAX とした ECXO について検討することとした。

本報告では、C 言語 [14] で EAX を実現し、主制御対象オペレータを EAX とした ECXO についてその有効性を世界共通データ TSPLIB の一つである 198 都市問題 d198 [15] によって検証したので報告する。

平成 19 年 12 月 17 日受理

* システム情報工学科・教授

2. 巡回セールスマン問題

巡回セールスマン問題 (TSP: Traveling Salesman Problem) とは「相互に結ばれている N 個の都市をセールスマンが一度ずつ訪問し最初の都市へ戻って来る巡回路のうち、総距離が最小となる巡回路を探索する問題」である。巡回路数は N 個の都市を訪れる順に並べる組み合わせ数分あり逆順を同一とみなすと $(N-1)!/2$ 通りと計算できる。例えば、 $N=20$ の時、 6.1×10^{16} と計算され、1 秒間に 100 万の経路長を計算できるコンピュータで 1,000 年以上の計算時間となる。従って、 N が 20 以上と大きくなるとコンピュータでも計算が困難となる。このような膨大な組み合わせの中から少ないコンピュータ時間で最適解を探索する手法を発見することが TSP の課題である [15]。

3. 検討の位置づけ

これまで、TSP の解法として、(1) 生物進化のメカニズムを遺伝子交叉や突然変異で擬似する遺伝的アルゴリズム (GA: Genetic Algorithms)、(2) 蟻の群行動の学習メカニズムを擬似するアアントコロニー最適化手法 (ACO: Ant Colony Optimization) [13]、(3) 物質を高温から低温に徐々に下げていき安定した素材を得る焼きなまし手法を擬似するシミュレーテッドアニーリング手法 (SA: Simulated Annealing) [12]、(4) 巡回路を幾つか (r 個: $r \geq 2$) に分解し分解した部分巡回路の順番や方向を入れ替える Lin-Kernighan 法 [11] が提案されている。各手法単独では局所最適解に陥り大域的に最適解を探索することが困難である。

TSP の課題は巡回路の多様性を確保しつつ大域的な解を探索することである [9]。すなわち、局所的に最適で異なった巡回手順や巡回路長を持つ巡回路を混在させて大域的な解を探索することである。我々はこれまで TSP の最適解の近似を効率的に得る新しい手法として先に

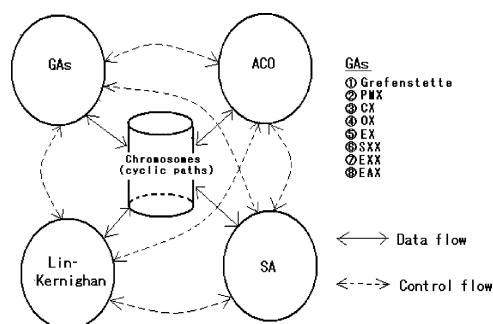


図1 拡張遺伝子交叉オペレータ交代法 ECXO 中の EAX の位置づけ

述べた GAs, ACO, Lin-kernighan, SA の各遺伝子交叉オペレータを任意の時点で交代可能な拡張遺伝子交叉オペレータ交代法 (ECXO: Extended Changing Crossover Operators) を検討してきた。GAs は更に、(A) 順列の互換やその表現方法に着目した Grefenstette [3], PMX [2], CX [4], OX [5], (B) 各都市を結ぶ枝の長さや枝の繋がり方に着目した EX [6], SXX, EXX, EAX に分類できる。ECXO [17] の実現方式を図1に示す。ECXO は、各遺伝子交叉オペレータが探索した巡回路情報を Chromosomes という遺伝子ファイルに保管し、他の遺伝子交叉オペレータがそれを参照引き継ぎ更新可能とする。これにより、巡回路の多様性を確保しつつ広域的な解の探索を可能とする。

一方、巡回路の多様性を確保しつつ大域的な解を探索する単独の手法として枝組み立て交叉 EAX が有効であることが知られており、我々もその有効性を C プログラミング実験で検証できた。本報告では、ECXO の検討の一貫として主制御対象オペレータを EAX とした ECXO について検討経過を報告する。

4. EAX のアルゴリズム

4.1 基本的考え方

TSP を遺伝的アルゴリズムで実現する場合、(i) 両親の優れた形質を子に継承すること、

(ii) 両親から生成される子のバリエーションをできるだけ保つことが重要である。この2つの要求は競合するので、そのバランスをとりながら最適解を探索する手法が望まれている。従来の EXX や SXX は上記の (i) の条件は満たすが、(ii) の条件を満たさなかったため、解への収束効率は高いものの、局所最適解に陥り易かった。EAX は条件 (i) (ii) を、バランスをとって満たしながら最適解に到達する方法として開発された。

EAX は巡回路を都市と都市を結ぶ枝の集合として捉える。この枝が交叉や突然変異の対象となる遺伝子である。EAX は親 A と親 B の同数の枝を交換して子供を生成する。親 A の枝情報と親 B の枝情報を和した枝情報から親 A の枝 (都市と都市を直接結ぶ路) を順方向に親 B の枝を逆方向に交互に辿って巡回路 (A-B サイクル) を発見し、それ等を相互に交換して局所的に最適な部分巡回路群を構成する。そして部分巡回路間の距離が最短になるように部分巡回路を結合して大域的に最適な巡回路を再構成する。そのような方法で EAX は親 A の有効な枝情報を子に引き継ぐと共に、発見する A-B サイクル数や A-B サイクルで交換する両親の枝を多くすることにより、子のバリエーションを保っている。EAX は、局所的に最適となっている両親の部分巡回路を結合して、更に短い巡回路を構成する従来の EXX や SXX の拡張となっている。

4.2 本研究での EAX の実現方式の特徴

(1) A-B サイクルの構成法

EAX の論文では無向グラフ (STSP: 対称 TSP) と有向グラフ (ATSP: 非対称 TSP) のそれぞれに対して A-B サイクルの構成法を示している。本研究では STSP に適当な向きをつけ ATSP に問題を変換して A-B サイクルを構成した。従って、本検討での ATSP では、都市 A から都市 B に向かう有向枝 $A \rightarrow B$ の長さ $\text{length}(A \rightarrow B)$ と都市 B から都市 A に向かう

有向枝 $B \rightarrow A$ の長さ $\text{length}(B \rightarrow A)$ は等しい。

(2) 緩和個体の構成法

k 個の A-B サイクルが生成された場合、どの A-B サイクルを用いて両親の枝を交換するかを選択の仕方により、巡回路 A ならびに巡回路 B から生成可能な緩和個体数は 2^k となる。本研究では生成された各々の A-B サイクルのみを利用して巡回路 A ならびに巡回路 B から緩和個体を生成する方式とした。従って巡回路 A ならびに巡回路 B から、各々 k 個、合計 $2k$ 個の子供の巡回路が生成される。そのうち巡回路長の最も短かな2個の巡回路を子供として選択する方式とした。

(3) 世代交代方式

ルーレット方式 (または、親 A は現世代の全ての個体とし、親 B をランダムに選択する) 等の方法により両親を2個選択する。(2) で述べたように、両親は、EAX により遺伝子交叉を行い生成した $2k$ 個の子供の中から、巡回路長の最も短い子供を2個体残す。その子供を両親として更に孫を EAX で2個体生成する。この方法で Ncross 回 EAX による交叉を繰り返し、子供を生成する。EAX による交叉により両親と同一個体が生成されるのみで、探索する巡回路に変化がみられなくなったら交叉は中止する。2個の親と生成した2個の子供の間で巡回路長を比較しその中で最も巡回路の短い個体2個体を次世代の子供として残す方式とした。これはエリート戦略と呼ばれる個体選択法の1つである。

4.3 EAX のアルゴリズム

以下の手順で親 A と親 B から子供 C と子供 D を生成する。

(1) 親巡回路の枝集合による表現

親 A と親 B の巡回路に巡回方向を適当に付与する。親 A と親 B の巡回路を枝 (現在訪れている都市と次に訪れる都市) の集合で表現する。また、二都市間を結んでいる枝の長さはユーク

リッド空間の場合、ピタゴラスの定理等により事前に測定しておく。

(2) A-B サイクルの構成

親 A の枝 (都市と都市を直接結ぶ路) を順方向に親 B の枝を逆方向に交互に辿って閉じた小道を構成する。この小道を A-B サイクルと呼び、辺の重複を許さない点の重複のみを許す開始点と終了点が一致した経路である。尚、A-B サイクルでは、それを構成する A と B の枝の数は一致しなくてはならない。また、同じ枝であっても両親が異なれば別の枝として区別される。

一般に親 A と親 B から複数の A-B サイクルができる。

この時、親 A と親 B を構成する全ての枝はいずれかの AB サイクルに含まれており重複しないこと、親 A と親 B から唯一つの A-B サイクル群が構成できることがわかっている。

(3) 緩和個体の生成

A-B サイクルを構成する枝を交互に交換して緩和個体を構成する。

A-B サイクルの枝は「親 A の枝 e_{A1} 」, 「親 B の枝 e_{B1} を逆順にした枝」, 「親 A の枝 e_{A2} 」, 「親 B の枝 e_{B2} を逆順にした枝」… という順番で並んでいるが、これを親 A の枝 e_{A1} と親 B の枝 e_{B1} を交換する、親 A の枝 e_{A2} と親 B の枝 e_{B2} を交換するという風に親 A と親 B が交互にその枝を交換して緩和個体を生成する。

緩和個体は親 A と親 B からそれぞれ生成され、一般に幾つかの部分閉路群から成る。

(4) 緩和個体を構成する部分閉路を最短枝でつなぎ合わせて巡回路を再生成する。

以下の方法で巡回路を再構成する (茨木著文献 [9] を参照のこと)。

- ① 部分閉路群の中から、部分閉路を構成する都市数が最も少ない部分閉路 X を選択する。
- ② 以下の手順で、部分閉路 X と部分閉路 Y を併合することにより部分巡回路長の削減量が最も大きくなるように部分閉

路 Y を選択する。これは、図形的には①で選択した部分閉路 X との距離 r が最も近い部分閉路 Y を併合して新たな部分閉路 Z を構成することを意味する。

(i) 部分閉路 X, Y の併合による新たな部分閉路 $Z (X, Y)$ の生成

部分閉路 X からある枝 $e = u \rightarrow v = [u, v]$ を取り除いた枝集合を $X[\bar{e}]$ とする。部分閉路 Y からある枝 $e' = u' \rightarrow v' = [u', v']$ を取り除いた枝集合を $Y[\bar{e}']$ とする。新たな枝 $e'' = u \rightarrow v' = [u, v']$ と $e''' = u' \rightarrow v = [u', v]$ を作成する。枝集合 $X[\bar{e}]$ と枝集合 $Y[\bar{e}']$ と枝 e'' と枝 e''' を和した枝集合を $Z' = Z'(X[\bar{e}], Y[\bar{e}'])$ とする。すると $Z' (X[\bar{e}], Y[\bar{e}'])$ は、部分閉路 X と部分閉路 Y を併合した新たな部分閉路となる。このような閉路 $Z' (X[\bar{e}], Y[\bar{e}'])$ のうち、部分巡回路長が最も短くなるように枝 e と e' を選び、その併合閉路を $Z (X, Y) = \min_{(e, e')} Z' (X[\bar{e}], Y[\bar{e}'])$ で示す。巡回路長の減少量 $\Delta(e, e')$ は $\Delta(e, e') = (\text{length}(e) + \text{length}(e')) - ((\text{length}(e'') + \text{length}(e'''))$ で計算できる。この減少量の最大値を満たすように部分閉路 X の枝 e と部分閉路 Y の枝 e' を選ぶ。この減少量の逆数相当が部分閉路 X と部分閉路 Y の距離 r となる。尚、部分閉路 Y の逆順とした部分閉路 \underline{Y} に対しても同様に $Z (X, \underline{Y})$ を求める。

(ii) 部分閉路 X に対して巡回路長の減少量が最も多くなるように部分閉路 Y (または \underline{Y}) を選択する。

③②で選択した部分閉路 Y (または \underline{Y}) と部分閉路 X を併合し新たな部分閉路 Z を生成する。

④ たった一つの閉路となるまで①～③を繰り返す。

(5) (4) で再構成した全ての個体のうち最も経路長の短い 2 個体を次世代の子候補として残す。

4.4 例題

EAX アルゴリズムの適用事例を述べる。

図 2 にその詳細を示す。

4.4.1 例題データ (図 2 (A))

3 次元ユークリッド空間上の 6 都市から成る巡回路を考える。各都市は、都市番号 1, 2, 3, 4, 5, 6 の順番に以下の {緯度, 経度} に位置する。

六都市データの緯度, 経度

C 言語で以下のように表現される。

```
static struct {double x; double y;} cityxy
[CityNo]=
```

```
{ {40.68333, 141.3833} /* 1: misawa*/,
  {35.55, 139.716} /*2: Tokyo*/,
  {34.883, 136.833} /*3: nagoya*/,
  {34.783, 135.483} /*4: osaka*/,
  {33.566, 130.433} /*5: fukuoka*/,
  {37.9166, 139.05} /*6: niigata*/};
```

4.4.2 EAX アルゴリズム適用プロセス

(1) 親巡回路の枝集合による表現

親巡回路 (図 2 (B))

遺伝子交叉の対象となる初期巡回路を辿る都市の順番に以下とする。

親 A $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 1$

親 B $5 \rightarrow 3 \rightarrow 1 \rightarrow 4 \rightarrow 2 \rightarrow 6 \rightarrow 5$

都市間の距離は地球の球面上の距離として求める。地球の半径を 6,378.1 km, 円周率 Π を 3.1415927 で計算すると, 親 A の経路長は 2,754.50 km 親 B の経路長は 3,784.19 km である。

親巡回路の枝集合による表現

枝は有向であり, 1 から 2 に向かう枝を $1 \rightarrow 2 = [1, 2]$ と表現する。また, 1 から 2 に向かう枝 $1 \rightarrow 2$ と 1 から 2 に向かう枝 $2 \rightarrow 1 = [2, 1]$ は区別する。

親 A の枝集合による表現: $[1, 2], [2, 3], [3, 4], [4, 5], [5, 6], [6, 1]$

親 B の枝集合による表現: $[5, 3], [3, 1], [1, 4], [4, 2], [2, 6], [6, 5]$

(2) A-B サイクルの構成 (図 2 (C))

親 A と親 B から以下の 2 つの AB サイクルができる。

A-B サイクル (1)

$1 \rightarrow 2 \leftarrow 4 \rightarrow 5 \leftarrow 6 \rightarrow 1 \leftarrow 3 \rightarrow 4 \leftarrow 1$

A-B サイクル (2) $2 \rightarrow 3 \leftarrow 5 \rightarrow 6 \leftarrow 2$

(説明) A-B サイクル (1) の最初の枝として親 A にて最初に訪れる都市 1 と二番目に訪れる都市 2 を結ぶ枝を選ぶ。それを $1 \rightarrow 2 = [1, 2]$ で表現する。次の $2 \leftarrow 4$ で親 B の枝 $4 \rightarrow 2 = [4, 2]$ を逆順に辿った枝であること, 次の $4 \rightarrow 5 = [4, 5]$ は親 A の枝であり, 次の $5 \leftarrow 6$ は親 B の枝 $6 \rightarrow 5 = [6, 5]$ を逆順に辿った枝であること, 次の $6 \rightarrow 1 = [6, 1]$ は親 A の枝であり, 次の $1 \leftarrow 3$ は親 B の枝 $3 \rightarrow 1 = [3, 1]$ を逆順に辿った枝であること, 次の $3 \rightarrow 4 = [3, 4]$ は親 A の枝であり, 次の $4 \leftarrow 1$ は親 B の枝 $1 \rightarrow 4 = [1, 4]$ を逆順に辿った枝である。こうして都市 1 を終始点とした閉じた小道が構成される。同様に, 親 A と親 B の残りの枝は A-B サイクル (2) を構成する。A-B サイクル (2) の $2 \rightarrow 3 = [2, 3]$ は親 A の枝であり, 次の $3 \leftarrow 5$ は親 B の枝 $5 \rightarrow 3 = [5, 3]$ を逆順に辿った枝であること, 次の $5 \rightarrow 6 = [5, 6]$ は親 A の枝であり, 次の $6 \leftarrow 2$ は親 B の枝 $2 \rightarrow 6 = [2, 6]$ を逆順に辿った枝である。

(3) A-B サイクルの枝を交互に交換して緩和個体を生成する。

(a) A-B サイクル (1) から以下の緩和個体が生成される。(図 2 (D))

① 親 A から生成される緩和個体は以下の 2 つの部分閉路から成る。

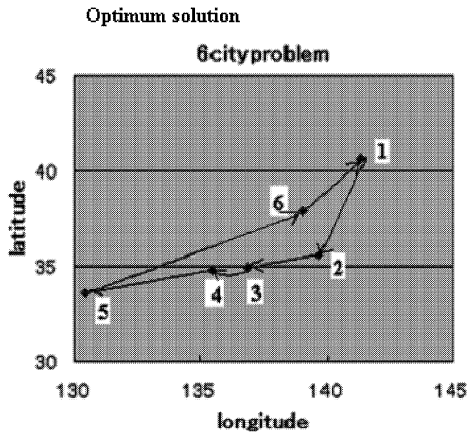
$4 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 4, 6 \rightarrow 5 \rightarrow 6$

② 親 B から生成される緩和個体は以下の 2 つの部分閉路から成る。

$5 \rightarrow 3 \rightarrow 4 \rightarrow 5, 6 \rightarrow 1 \rightarrow 2 \rightarrow 6$

(説明) 親 A の枝表現 $[1, 2]$ は親 B の枝表現 $[4, 2]$ と, 親 A の枝表現 $[3, 4]$ は親 B の枝表現 $[1, 4]$ と, 親 A の枝表現 $[4, 5]$ は親 B の枝表現 $[6, 5]$ と, 親 A の枝表現 $[6, 1]$ は親 B の枝表現 $[3, 1]$ と, それぞれ交換する。それ以

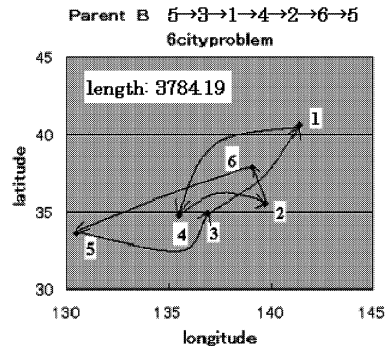
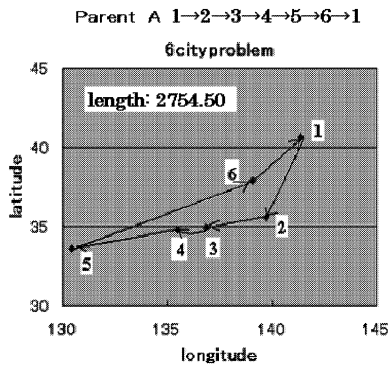
(A) テストデータと最適解



テストデータ

- ① {40.68333, 141.3833} /*1: misawa*/,
- ② {35.55, 139.716} /*2: Tokyo */,
- ③ {34.883, 136.833} /*3: nagoya */,
- ④ {34.783, 135.483} /*4: osaka*/,
- ⑤ {33.566, 130.433} /*5: fukuoka*/,
- ⑥ {37.9166, 139.05} /*6: niigata*/

(B) 親 A と親 B の巡回路



(C) 親 A と親 B から生成される A-B サイクル

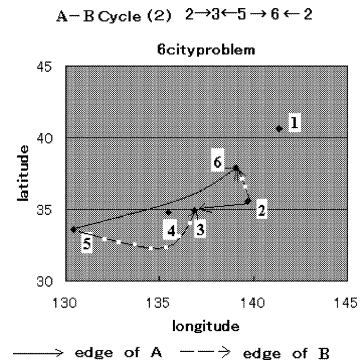
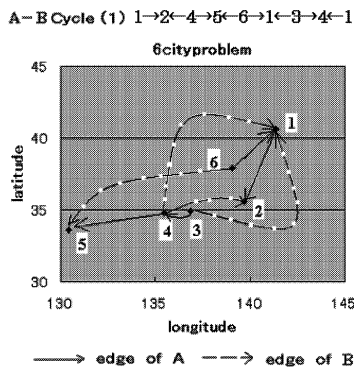


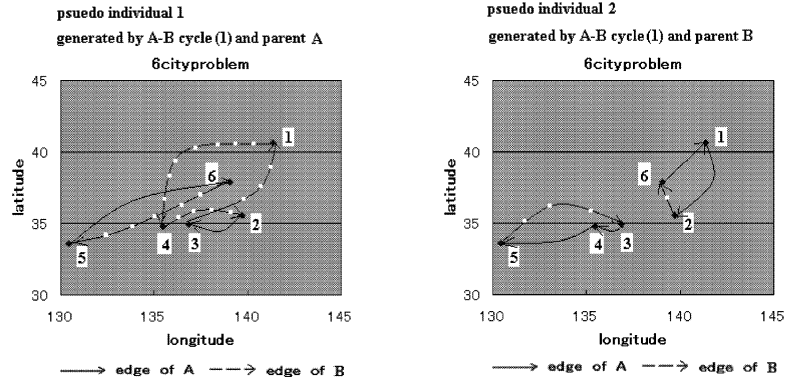
図 2 枝組み立て交叉の例題 (6 都市)

(A) テストデータと最適解

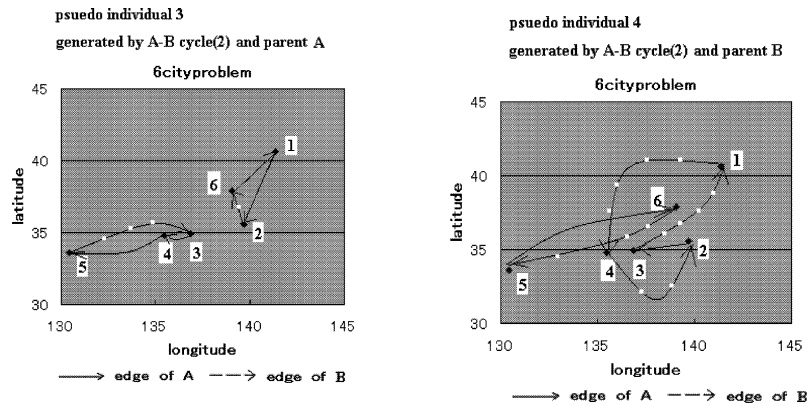
(B) 親 A と親 B の巡回路

(C) 親 A と親 B から生成される A-B サイクル

(D) A-B サイクル (1) から生成される緩和個体



(E) A-B サイクル (2) から生成される緩和個体



(F) A-B サイクル (1) から生成された緩和個体を修正してできる子供

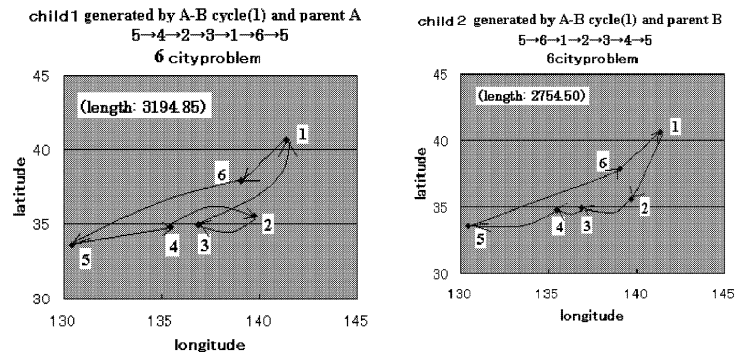


図2 枝組み立て交叉の例題 (6都市)

(D) A-B サイクル (1) から生成される緩和個体

(E) A-B サイクル (2) から生成される緩和個体

(F) A-B サイクル (1) から生成された緩和個体を修正してできる子供

外の枝については、親 A の場合は [2, 3], [5, 6], 親 B の場合は [5, 3], [2, 6] は変わらない。すると、元の枝の順番に親 A から [4, 2], [2, 3], [1, 4], [6, 5], [5, 6], 親 B から [5, 3], [6, 1], [3, 4], [1, 2], [2, 6], [4, 5] の緩和個体が生成される。これを整理して以上ようになる。

(b) A-B サイクル (2) から以下の緩和個体が生成される。(図 2 (E))

① 親 A から生成される緩和個体は以下の 2 つの部分閉路から成る。

1 → 2 → 6 → 1, 5 → 3 → 4 → 5

② 親 B から生成される緩和個体は以下の 2 つの部分閉路から成る。

2 → 3 → 1 → 4 → 2, 5 → 6 → 5

(4) 緩和個体を構成する部分閉路を最短枝でつなぎ合わせて巡回路を再生成する。

以下の 4 つの巡回路が合成される。

(a) A-B サイクル (1) から構成された緩和個体からの巡回路の合成 (図 2 (F))

① 親 A から生成された緩和個体 4 → 2 → 3 → 1 → 4, 6 → 5 → 6 から合成される巡回路

(子 1) 5 → 4 → 2 → 3 → 1 → 6 → 5 (経路長: 3,194.85)

② 親 B から生成される緩和個体 5 → 3 → 4 → 5 6 → 1 → 2 → 6 から合成される巡回路

(子 2) 5 → 6 → 1 → 2 → 3 → 4 → 5 (経路長: 2,754.50)

(b) A-B サイクル (2) からの緩和個体の構成 (図は略) 生成される子は A-B サイクル (1) と同じである。

① 親 A から生成された緩和個体 1 → 2 → 6 → 1, 5 → 3 → 4 → 5 から合成される巡回路

(子 3) 1 → 2 → 3 → 4 → 5 → 6 → 1 (経路長: 2,754.50)

② 親 B から生成された緩和個体 2 → 3 → 1 → 4 → 2, 5 → 6 → 5 から合成される巡回路

(子 4) 5 → 4 → 2 → 3 → 1 → 6 → 5 (経路長: 3,194.85)

(5) (4) で再構成した全ての個体のうち最

も経路長の短い 2 個体を次世代の子候補として残す。

以下の 2 個体が残る。

(子 1) 5 → 4 → 2 → 3 → 1 → 6 → 5 (経路長: 3,194.85)

(子 2) 5 → 6 → 1 → 2 → 3 → 4 → 5 (経路長: 2,754.50)

子 2 は親 A と同じ長さの巡回路, 子 2 は, 親 B よりも長さの短い巡回路となっており, 巡回路長の改善が見られる。

4.4.3 EAX と EXX ならびに SXX が生成する子の比較

親 A の枝 [1, 2] と親 B の枝 [1, 4] を交換することから交叉を始める EXX が生成する子の巡回路 3 → 4 → 1 → 2 → 6 → 5 → 3 (経路長: 3,809.75), 6 → 1 → 3 → 2 → 4 → 5 → 6 (経路長: 3,194.85) より EAX が生成した子の巡回路の方が, 巡回路長は短くなっている。更に, SXX では, 親 A と親 B の任意の都市を起点としてある長さの部分巡回路比較し構成する都市が同じとなる部分巡回路は親 A と親 B しかなく, 従って親 A と親 B と同じ子が生成される。尚, 最短巡回路は 5 → 6 → 1 → 2 → 3 → 4 → 5 であり, その経路長は 2,754.50 である。

5. C プログラム開発

C 言語の開発規模は以下の通りである。

コメントを除き, 遺伝的アルゴリズム本体部 22 関数約 1.6×KS (キロ (=10³) 行), EAX 部 19 関数約 1.0 KS (キロステップ), 合計 41 関数約 2.6 KS の C プログラムである。遺伝的アルゴリズム本体部は遺伝子交叉オペレータに依存しない部分である。その内訳を「関数とその機能, 規模, サイクロマチック数(条件文数+1)」として表 1 にまとめる。

表 1. EAX を実現する C 関数の機能, サイクロマチック数, 実行文行数

		関数名	機能	サイクロマチック数	実行文行数
本体部	1	srand	乱数種を設定する	1	4
	2	rand	一様乱数を発生させる	1	5
	3	App_Execute	遺伝子操作の実行を制御する	5	18
	4	App_Initialize	初期化処理を制御する	16	83
	5	App_Initialize_for_ACO	ACO の初期化をする	15	76
	6	initialize_population	初期人口の生成を制御する	6	30
	7	copy_population	次世代の子供を形成する	6	25
	8	update_population 2	部分的に次世代の子供を形成する	3	10
	9	selectParent	親を選択する	2	15
	10	next_generation	次世代の子の生成を制御する	87	405
	11	improve_crossover	2opt 法で個体を改良する	14	42
	12	improve_crossover_lin_kernighan_3opt	3opt 法で個体を改良する	41	148
	13	reverse	巡回路を逆にする	2	10
	14	dist_fitness	適応度を計算する	3	13
	15	statistic	統計データを出力する	20	76
	16	flip	乱数がある閾値を越えるかを判断する	3	8
	17	gen_root	初期の個体を形成する	6	22
	18	random_ga	0 から 1 の乱数を発生させる	2	9
	19	main	遺伝的アルゴリズムの主制御	130	422
	20	App_Execute_1	即時世代交代を制御する	4	16
	21	next_generation_1	即時に世代を交代する	17	70
	22	iterate_chromosome	即時型のエリート選択処理	24	71
EAX 部	1	create_edge	個体から枝を生成する	3	25
	2	reverse_eax	部分閉路の方向を逆転させる	5	19
	3	statistics_compact	簡易統計データを出力をする	1	8
	4	copy_edge	枝を複写する	2	11
	5	eliminate_non_efficient_ab_cycle	不要な AB サイクルを削除する	15	68
	6	create_ab_cycle	A-B サイクルを生成する	21	96
	7	rearrange	枝の順番を都市の訪問順に並べ替える	18	82
	8	create_psuedo_individual_1	親 A の緩和個体を生成する	7	38
	9	create_psuedo_individual_2	親 B の緩和個体を生成する	7	38
	10	union_loops	部分閉路と部分閉路を統合する	11	63
	11	re_generating_edge	枝を再構成する	6	31
	12	search_loop	部分閉路を探索する	2	18
	13	search_minimum_loop	都市数が最小な部分閉路を探索する	4	24
	14	re_generate_loop	部分閉路を再構成する	4	45
	15	modify_psuedo_individual	緩和個体を修正し個体を形成する	9	50
	16	edge_check	枝をチェックする	7	33
	17	create_individual	個体を枝から形成する	2	10
	18	crossover_eax	EAX 交叉処理を行う (本体部)	36	212
	19	crossover_eax_1	Ncross 回 EAX 交叉処理を行う	27	171
本体部合計				408	1,578
EAX 部合計				187	1,042
合計				595	2,620

6. 実験結果

6.1 実験データ

TSPLIB に掲載されている d198 の実験データを図 3 に示す。d198 は 2 次元ユークリッド空間であり、空間上大きく 2 群に分けられるのが特徴である。

6.2 実験環境

COMPAC NX6320 HP Genuine Intel (R) CPU 1.66 GHz, 504 MB RAM

6.3 評価対象遺伝子交叉オペレータと評価項目

(1) 評価対象遺伝子交叉オペレータ

以下の遺伝子交叉オペレータについて機能と性能を評価した。

- ① EXO (ACO- \rightarrow EXX- \rightarrow EAX)
- ② ECXO (EX- \rightarrow EXX- \rightarrow EAX)
- ③ ACO
- ④ EX
- ⑤ EXX
- ⑥ EAX
- ⑦ Lin-Kernighan
- ⑧ SA (2-opt+3-opt)

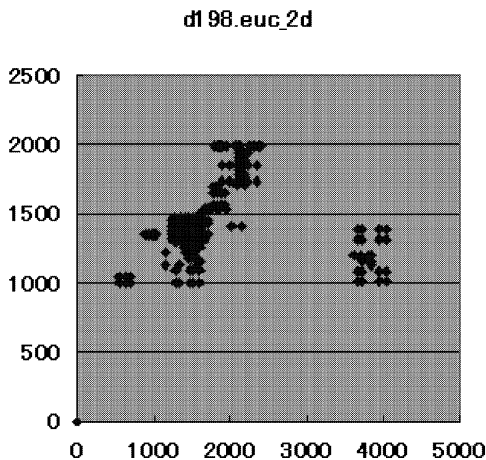


図 3 実験データ

⑥ EAX 以外の各手法の機能概要は以下の通りである。

①② 拡張遺伝子オペレータ交代法 (Extended Changing Crossover Operators) は任意の時点で遺伝子オペレータを交代させる方法である。① EXO (ACO- \rightarrow EXX- \rightarrow EAX) は ACO から EXX, EXX から EAX に交代させる方法である。同様に ② ECXO (EX- \rightarrow EXX- \rightarrow EAX) は EX から EXX, EXX から EAX に交代させる方法である。

③ アントコロニー最適化手法 (ACO: Ant Colony Optimization) では餌採集時の蟻の集団行動の知能をフェロモンで説明する。それを TSP に応用する。過去に旅をした蟻の経路と経路長に関する学習情報を、巡回路上の枝上にフェロモンとして残す。後続する蟻は枝上のフェロモン量と枝の短さを考慮して次に訪れる都市を確率的に選択する。

④ EX (Edge Recombination Crossover) は親の隣接都市リストの中から現在訪れている都市に最も近い都市を順番に選択していく方法である。

⑤ EXX (Edge Exchange Crossover) は両親の枝情報 (どの都市からどの都市に訪れるのか) を次の世代のいずれかの子供の枝情報として伝える方法である。出発点を同じとする両親の枝情報の交換から始めて枝の交換により生成した子供の経路が巡回路でなくなった場合は、部分巡回路を逆順にして交換した枝につなげる等の修正を繰り返して巡回路を再生成する方法である。EAX は EXX の拡張版である。

⑦ Lin-Kernighan 法では任意の巡回路に対する網羅的な 2 分割探索 (2-opt: 部分巡回路を逆順でつなぐこと) 及び 3 分割探索 (3-opt), そして確率的な r -opt 探索 ($r \geq 4$) が可能である。

⑧ シミュレーテッドアニーリング手法 (SA: Simulated Annealing) では r -opt 法 ($r=2, 3$) により現状の探索経路長より長い巡回路を探索した場合でも巡回路長の増加量が少

なければその状態に確率的に遷移する。これにより局所最適解からの脱出を図る。試行回数が増えるにつれ巡回経路長が長い状態への遷移確率は低くなり解は安定に向かう。

(2) 評価項目

- ① 探索した最短経路長
- ② 最短経路長を探索するに要した CPU 時間
- ③ 最短経路長を探索するまでに要した遺伝子交叉回数 (旅の数)

6.4 各種遺伝子交叉オペレータの実行パラメータ

表 2 に実行パラメータと d198 でのパラメータ値をまとめる。GAs の実行パラメータについては EAX の実行パラメータでまとめている。観測世代数を除き, EX, EXX も EAX と同じパラメータである。

6.5 最適解

図 4 に EAX や ECXO が探索した経路長 15,780 の最適解を示す。

6.6 実験結果

表 3 に実験結果をまとめる。

表 3 では, 遺伝子交叉オペレータ毎に探索した最短経路長, 最短経路を探索するのに要した CPU 時間 (sec) と遺伝子交叉回数 (世代数×人口数) とその各々の内訳をまとめている。

以下に実験結果をまとめる。

(1) 人口数 1,000, 遺伝子交叉確率 0.8 の EAX 法により 1,184 秒 14 世代目に最短経路長 15,780 を探索した。当数値は, TSPLIB で公開されている最短経路長と一致している。

(2) EX 法, EXX 法, ACO 法, Lin-Kernighan (4opt), SA (2opt+3opt) 単独ではそれぞれ 15,895, 16,428, 16,570, 15,976, 17,560 の経路長しか探索できなかった。

(3) ACO 法から EXX 法に交代し, その後に EAX に交代させる ECXO 法 (ACO->

EXX->EAX) では EAX 単独で最適解を探索するより短い時間 884 秒 6,724 回の遺伝子交叉回数で最短経路長 15,780 を探索できた。最適解を探索するのに要したコンピュータ時間と遺伝子交叉回数の内訳を表 3 に示す。

(4) EX 法から EXX 法に交代し, その後に EAX に交代させる ECXO (EX->EXX->EAX) では EAX 単独で最適解を探索するより短い時間 708 秒 296 世代で最短経路長 15,780 を探索できる。

6.7 検討課題

(1) ECXO (EX->EXX->EAX) の性能向上策

EXO から EXX への世代交代時期について, EX 法単独で経路長 15,895 を探索した 408 秒 133 世代目 (1 世代は 1,000 個の個体から成る) に EXX に交代させると, EXX はその 433 秒後 6,117 世代目に経路長 15,820 を探索したが, その後 EXX から EAX に交代させても EAX は更に短い経路長を探索することはできなかった。しかし, EX から EXX への交代時期を 121 秒 35 世代目早めると EXX はその 20 秒後 254 世代目に経路長 15,841 を探索しその後 EAX に交代させると 567 秒後に EAX は最適解 15,780 を探索できた。この結果は, EX によって部分的に最適解となっているが経路長にばらつきがある EX の比較的初期の段階, 解に多様性がある状態で, 局所最適な部分枝をつなげて広域的に最適解を探索する EXX そして EAX に交代させる方式が解の探索効率を向上させるのに有効であることを示唆している。どの世代が最適な EX から EXX への交代時期なのかは今後の課題である。

(2) ECXO (ACO->EXX->EAX) の性能向上策

現 ECXO (ACO->EXX->EAX) では ACO が最適解を探索した 614 秒目 84 世代に ACO から EX に交代し, その後 87 秒後 6,440 世代目に EXX は経路長 15,962 を探索し, その後

表 2. GAs と ACO の実行パラメータ

(1) 遺伝的アルゴリズム (GAs) の実行パラメータと d198 の場合のパラメータ値

NO	実行パラメータ	d198
1	遺伝子交叉オペレータ 1	EAX
2	交叉オペレータを交代後の遺伝子交叉オペレータ 2	—
3	遺伝子交叉オペレータの交代時期 遺伝子交叉オペレータの交代時期を区間 (開始と終了) で指定する。 開始と終了の区間の世代で遺伝子交叉オペレータを交代させ GAs を実行する。	—
4	一括型世代交代方式または即時型世代交代方式の選択	一括型
5	2opt 法使用の有無 ・遺伝子交叉で生成した子供に 2opt 法を 1 回適用して、探索距離の改善を図る	YES
6	2opt 法使用の場合、平野 2opt 法とするか	NO
7	生成した子供を間引きするか否か？ YES の場合適用度の高い方の子のみ残す。	YES
8	統計出力の種別 (各世代情報の標準出力への出力の有無)	YES
9	3opt 法使用の有無 ・遺伝子交叉で生成した子供に 3opt 法を 1 回適用して、探索距離の改善を図る。2-opt 法を適用した後 に 3opt 法を適用して距離を改善を図る。	YES
10	人口数	1,000
11	最大観測世代数 (最適)	20(14)
12	統計出力タイミング: 人口数の約数分のタイミング等で部分的に世代を更新する場合、何人の子供が 生成されたら、部分的に世代を更新するのかを指定する。	1,000
13	ユークリッド空間において、二点間の距離を小数点以下を四捨五入した整数解で求めるか	YES
14	ATT タイプ (擬ユークリッド空間。距離を 1/10 に縮めて測定) で二点間の距離を測定するか	NO
15	エリート再生成処理エリート親 A と親 B から子供を数個体生成する。生成した子供の中から適応度 の高い子供二個体を、改めて親 A と親 B として、子供を生成する。このような交叉処理を Ncross 回 行い子供を生成するか? (EAX のみ有効)	YES
16	エリート再生成処理における Ncross 回数 (EAX のみ有効)	20
17	ルーレット方式で親を選択するか否か (EAX のみ有効)	NO
18	ルーレット方式で親を選択しない場合、親 A として現世代の個体全てが対象となる。もう一方の親 B を、現世代の個体から任意に確率的に m 個選択して遺伝子交叉を行い、個体を $2m$ 個生成する。そ の中で優れた個体を 2 個残す。その際の m を指定する。(EAX のみ有効)	1
19	乱数種	1
20	遺伝子交叉確率	0.8
21	突然変異の有無	OFF
22	突然変異確率	—
23	親と子供を併せた個体群 α の間でトーナメントを行うか否か。YES なら個体群 α の中で適応度の高 い個体を残す。NO なら、親は残さないで子供のみを残す。	YES

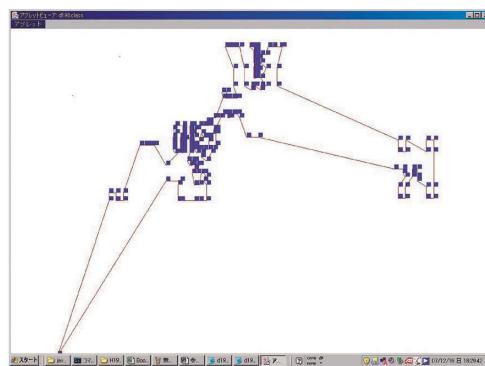
(2) ACO の実行パラメータと d198 の場合のパラメータ値

NO	実行パラメータ	d198
1	2 都市間の距離を一般ファイル出力するか？	NO
2	2 都市間の距離を一般ファイル入力し、隣接都市情報リストを作るか？	YES
3	次に訪問する都市を隣接する m 個の視隣接都市の中から優先的に次に訪れる都市を選択する時の m を指定する。	20
4	一世代の人口数	198
5	最終世代の人口のうち何匹の蟻の旅情報を GAs に引き継ぐか。	100
6	観測最大世代数 (最適)	100 (84)
7	蟻の旅に関する情報の統計出力周期	198
8	蟻の固定的な旅情報から初期フェロモン量を生成するか否か？	NO
9	都市 i と都市 j 間のフェロモン量における距離の重みづけ	2
10	フェロモン忘却率 1	0.3
11	フェロモン忘却率 2	0.3
12	フェロモン更新タイミング q : 何匹の蟻の旅度にフェロモンを更新するか	198
13	都市間の固定的リンクを有効にする世代数	2,000
14	フェロモン忘却率を 1 から 2 に変更する世代数	2,000
15	初期フェロモン量の重み	1
16	フェロモン量を初期データとして引き継ぐか	NO
17	フェロモンによる SA の有無: エリート蟻のフェロモン量の重み α を時間によって少しずつ増加させるか	YES
18	エリート蟻のフェロモン量の重み α	2
19	フェロモン量の重み α を世代番号で更新するかフェロモン更新回数で更新するか？	世代番号
20	フェロモン量の重み α を周期的に更新するか否か？	YES
21	各蟻が始めに訪れる都市番号をランダムに決めるか？	NO
22	各蟻が二回目以降の旅で始めに訪れる都市番号をランダムに決めるか？	NO
23	探索経路を 2-opt 法と 3-opt 法を適用して確率的に変更・改善する SA (シミュレーテッド・アニーリング) を実現する際、距離の短縮度合いとこれまでの経過時間を考慮したボルツマンマシンの受理関数により確率的に次の状態に移行判断する。その制御パラメータである温度を世代番号で低くするか、フェロモン更新回数で低くするか指定。本 SA では経路長が短くなった時に次状態に確率的に遷移する。そうでない場合は遷移を保留する。	世代番号
24	2opt 法による SA (シミュレーテッド・アニーリング) を実現するか否かの指定。YES の場合、2opt 法を 1 回適用して距離を改良する。	YES
25	3opt 法による SA (シミュレーテッド・アニーリング) を実現するか否かの指定。YES の場合、3opt 法を 1 回適用して距離を改良する。2opt 法の後に 3opt 法。	YES
26	フェロモン量を更新時、更新期間で旅をした q 匹の蟻のうち最短経路を探索した蟻の経路情報のみでフェロモン量を更新するか否か？	NO
27	フェロモン量を更新時、最低のフェロモン量を保証するか否か	YES
28	固定的に接続する二都市の数と都市番号	0
24	乱数種により初期に訪れる都市番号が変わると共に ACO の実行結果も異なってくる。開始の乱数種と終了の乱数種を指定する。	1
25	ユークリッド空間で二点間の距離を小数点以下四捨五入した整数解とするか	YES
26	ATT タイプ (擬ユークリッド空間。距離を 1/10 に縮めて測定) で二点間の距離を測定するか否か	NO

表 3. 遺伝子交叉オペレータの評価結果

遺伝子交叉オペレータ	探索した 最短経路長	最短経路を探索 するのに要した CPU 時間 (sec)	最短経路を探索するのに 要した遺伝子交叉回数	
			世代数	人口数
ECXO (ACO-→EXX-→EAX)	15,780 内訳 ACO 16,570 → EXX 15,962 → EAX 15,780	884 内訳 ACO 614 → EXX 87 → EAX 183	6,724 内訳 ACO 84 → EXX 6,440 → EAX 200	198
ECXO (EX-→EXX-→EAX)	15,780 内訳 EX 16,256 → EXX 15,841 → EAX 15,780	708 内訳 EX 121 → EXX 20 → EAX 567	296 内訳 EX 35 → EXX 254 → EAX 140	1,000
EX	15,895	408	133	1,000
EXX	16,428	1,023	9,365	1,000
ACO	16,570 (15,867)	614 (17,066)	84 (107)	198 (1,980)
EAX	15,780	1,184	14 (×20*)	1,000
Lin-Kernighan (4opt)	15,976	427		1
SA (2opt+3opt)	17,560	46,564		1

*Ncross 回数



(朱線 : 最短経路)

図 4 d198 最適解

EXX から EAX に交代させて 183 秒後 200 世代で EAX は最適解 15,780 を探索した。(2) と同様な議論により ACO から EXX への交代時期を更に早めることにより更に CPU 時間を短

縮できると推察できる。最適な ACO から EXX への交代時期については今後の課題とする。

(3) EAX の性能向上

現在 ATSP (非対称 TSP) に変換して問題を解いているが、「STSP (対称 TSP) 方式での A-B サイクル決定法」や「最適な Ncross 回数や人口数の決定法」等を今後の課題とする。尚、TSPLIB の 532 都市問題 att532 や 575 都市問題 rat575 等の TSP データに対しても EAX は最適解を探索できることを C 実験で確認した。

(4) ACO の性能向上

ACO 単独では人口数 1,980, フェロモン更新タイミング 198 匹, 忘却率の時間による変更, 「距離とフェロモンによる SA 機能あり」のもとで, 15,867 の最適解を探索している。これ等のパラメータの設定法や EX で探索した解を初期値とした場合の実験等を今後の課題とする。

(5) Lin-Kernighan の性能向上

Lin-Kernighan は r -opt ($r \leq \text{city}/2$) が可能である。実験では $r=4$ としている。最適な r の探索法は今後の課題とする。またその初期値はランダムな値としている。(4)と同様, EX で探索した解を初期値とした場合の実験を今後の課題とする。

(6) SA (2opt+3opt) の性能向上

SA (2opt+3opt) の初期値はランダムな値としている。(4)と同様, EX で探索した解を初期値とした場合等を今後の課題とする。

7. おわりに

EAX を主制御オペレータとする拡張遺伝子オペレータ交代法 ECXO の有効性を, TSPLIB の 198 都市問題 d198 を用いた C プログラム実験で検証した。d198 について C プログラミング実験では ACO や EX, EXX 単独では最短経路を探索できなかったが, EAX では探索できた。EAX の課題の 1 つは, 局所的に最適となっている経路を形成するまでに時間がかかることである。d198 を用いた我々の実験では, EAX 単独に比較して ACO (または EX) から EXX に交代させた後に EAX を適用して最短路を探索する ECXO (Extended Changing Crossover Operator) が EAX の性能と機能を向上させるのに有効であることがわかった。EAX 等を主制御オペレータとした ECXO について, 更に実験空間を拡張して有効性を検証する予定である。

参考文献

- [1] J.H. Holland: Adaptation in Natural and Artificial Systems, MIT Press, 1992
- [2] D.E. Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc., 1989
- [3] J. Grefenstette, R. Gopal, B. Rosmaita, and D. Van Gucht: "Genetic Algorithms for the Traveling Salesman Problem", Proc. of 1st Int. Conf. on Genetic Algorithms and Their Applications, pp. 160-168 (1985)
- [4] I.M. Oliver, D.J. Smith, and J.R.C. Holland: "A Study of Permutation Crossover Operations on the Traveling Salesman Problem", Proc. of 2nd Int. Conf. on Genetic Algorithms, pp. 224-230 (1987)
- [5] L. Davis: "Applying Adaptive Algorithms to Epistatic Domains", Proc. of 9th Int. Joint Conf. on Artificial Intelligence, pp. 162-164 (1985)
- [6] D. Whitley, T. Starkweather and D'Ann Fuquary: "Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operation," Proc. of 3rd Int. Conf. on Genetic Algorithms, pp. 133-140 (1989)
- [7] M. Yamamura, I. Ono and S. Kobayashi: "Emergent Search on Double Circle TSPs using Subtour Exchange Crossover", Proc. of 1996 IEEE Int. Conf. on Evolutionary Computation, pp. 535-540 (1996)
- [8] K. Maekawa, N. Mori, H. Tamaki, H. Kita and Y. Nishikawa: "A Genetic Solution for the Traveling Salesman Problem by means of a Thermodynamical Selection Rule", Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC '96), pp. 529-534 (1996)
- [9] 永田裕一, 小林重信: "巡回セールスマン問題に対する交叉: 枝組み立て交叉の提案と評価", 人工知能学会誌, Vol. 14, No. 5, pp. 848-859, 1999.
- [10] M.-K. Tsai, J.-M. Yang, Y.-F. Tsai, C.-Y. Kao: "Some issues of designing genetic algorithms for traveling salesman problem, Soft Computing," No. 8, pp. 689-697, 2004.
- [11] S. Lin & B.W. Kernighan: "An Effective Heuristic Algorithm for the Traveling-Salesman Problem," Oper. Res. 21, pp. 498-516 (1973).
- [12] E. Aarts and J.K. Lenstra: Local Search in Combinatorial Optimization, Princeton University Press (2003)
- [13] M. Dorigo and T. Stutzle: Ant Colony Optimization, The MIT Press, 2004.
- [14] B.W. Kernighan and D.M. Richie: The C Programming Language, Second Edition, Bell Telephone Laboratories (1988)
- [15] 岩間一雄: 「アルゴリズム理論入門」, 昭晃堂, pp. 119-152 (2001)

- [16] R. Takahashi: "Solving the Traveling Salesman Problem through Genetic Algorithms with Changing Crossover Operators", in Proceedings of Fourth International Conference on Machine Learning and Applications, pp. 319-324, published by IEEE Computer Society (2005)
- [17] R. Takahashi: "A Performance Improvement of Solving the Traveling Salesman Problem through Uniting Changing Crossover Operators to Ant Colony Optimization", Advance in Natural Computation and Data Mining, Proceedings the 2nd International Conference on Natural Computation and the 3rd International Conference on Fuzzy Systems and Knowledge Discovery, pp. 114-130, Xidian University (2006)