巡回セールスマン問題を解く
拡張遺伝子交叉オペレータ交代法

高　橋　良　英*

Extended Changing Crossover Operators to Solve the Traveling Salesman Problem

Ryouei TAKAHASHI*

Abstract

In order to efficiently obtain an approximate solution of the traveling salesman problem (TSP), extended changing crossover operators (ECXOs) which can substitute any crossover operator of genetic algorithms (GAs) and ant colony optimization (ACO) for another crossover operator at any time is proposed. In this investigation our ECXO uses both EX (or ACO) and EXX (Edge Exchange Crossover) in early generations to create local optimum sub-paths, and it uses EAX (Edge Assembly Crossover) to create a global optimum solution after generations. With EX or ACO any individual or any ant determines the next city he visits from lengths of edges or tours' lengths deposited on edges as pheromone, and he generates local optimum paths. With EXX the generated path converges to a provisional optimal path. With EAX a parent exchanges his edges with another parent's ones reciprocally to create sub-cyclic paths, before restructuring a cyclic path by combining the sub-cyclic paths with making distances between them minimum. In this paper validity of ECXO is verified by our C experiments using medium-sized problems in TSPLIB, and it is shown that ECXO can find the best solution earlier than EAX.

Keywords: TSP, ECXO, GA, ACO, EAX

1. はじめに

巡回セールスマン問題（TSP: Traveling Salesman Problem)[4),21)] の最適解の近似を効率的に得る新しい手法として遺伝的アルゴリズム(GAs: Genetic Algorithms)[3),4)] とアーントコロニー最適化法（ACO: Ant Colony Optimization)[1),2)] 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)[22),23),24),25)] を提案する。ECXO では，ある遺伝子交叉オペレータが探索した巡回路情報を，他の遺伝子交叉オペレータが任意の時点で引継ぎ更新可能である。これにより，巡回路の多様性を確保しつつ広域的に最適な解の探索を可能とする。

本研究では，巡回路の多様性を確保しつつ広域的に最適な解の探索をする手法として近年着目を浴びている枝組み立て交叉（EAX: Edge Assembly Crossover)[11),12)] を主制御対象の遺伝子交叉オペレータとする ECXO を検討した。EAX では巡回路を都市と都市を結ぶ枝の集合としてとらえる。そして，両親の枝を交互に交換して部分巡回路群を生成した後，部分巡回路間の距離が最小となるように部分巡回路を結合して巡回路を再構成する。EAX は，両親の枝の

平成 20 年 12 月 15 日受理
* システム情報工学科・教授

交換と部分巡回路の逆順操作により子供の巡回路を生成する枝交換交叉（EXX : Edge Exchange Crossover）[9] の機能の拡張である。

我々の C プログラミング実験によれば，(i) EAX は両親の巡回路の部分経路長が最短となっていればその部分経路を結合して大域的に最適な解を生成する能力が枝再構成交叉（EX : Edge Recombination Crossover）[8] や ACO より優れていること。しかし，部分巡回路を生成する際に枝の長さに関する情報を利用しないため EAX 単独では局所的に最適な解を生成するのに時間がかかること，(ii) EX または ACO は両親の枝の長さに関する情報やフェロモンとして各枝に残された過去の旅の経路長に関する情報から子供が次に訪れる都市を決定するため，生成した経路は局所的に最適となっている。しかし大域的な枝と枝のつながりに関する情報から子供が次に訪れる都市を決定できないため局所最適解に陥りやすいことがわかっている。このため，本実験では比較的早期の世代では遺伝子交叉オペレータ EX または ACO ならびに EXX により局所的に最適で多様な解を生成し，これを入力として後期の世代では遺伝子交叉オペレータ EAX により大域的に最適な解を効率的に生成する ECXO を研究することとした。当 ECXO の有効性を，中規模 TSPLIB データ [26] を用いた C プログラミング実験で検証したので報告する。

2. 巡回セールスマン問題

あるセールスマンが n 箇所の都市を 1 回ずつ訪問して出発都市に戻ってくる巡回経路のうち，所要距離（または所要費用や所要時間）が最小となる経路を求める問題を巡回セールスマン問題（TSP）という。但しどの 2 都市間も直接辺で結ばれており，各辺の長さはわかっている。この時，逆順を同一経路と見なして全経路数 N は $(n-1)!/2$ 通りと計算できる。$n!$ は Stirling の公式により，$\sqrt{2\pi}\exp(-n)*n^{(n+1/2)}$ と近似されるが，この式は網羅的探索で最適解を探索するのに指数オーダ以上の時間がかかることを示している。例えば，$n=20$ の時，$N \fallingdotseq 6.1\times10^{16}$ 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実効上計算が困難となる。このような意味で TSP は n に関する多項式時間で解くことが困難な NP クラスに属する問題であると理解されている [16],[17]。

3. 拡張遺伝子交叉オペレータ交代法

3.1 背景

TSP の解法として，(i) 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），(ii) 蟻の採集行動における群知能の学習メカニズムを擬似するアーントコロニー最適化手法（ACO : Ant Colony Optimization），(iii) 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）[16]，(iv) 巡回路を幾つか（r 個 : $r>=2$）に分解した部分巡回路の順番や方向を入れ替えて更に短い巡回路を再生成する Lin-Kernighan 法 [13],[14] が提案されている。しかし，我々のこれまでの C プログラミング実験によれば，各手法単独では局所最適解に陥り大域的に最適な解を探索することが困難なことがわかっている。

TSP を GAs で解く時の課題は，巡回経路の遺伝子型を訪れる順番で並べた都市番号列で表現した時，単純な一点交叉や二点交叉等の遺伝子操作では致死遺伝子が生成されてしまう点である。致死遺伝子とは「n 箇所の都市を一回ずつ訪問して出発都市に戻ってくる」という TSP の解の条件を満たさない遺伝子のことであり，同じ都市を二度訪問したり訪れない都市のある経路を現わす。致死遺伝子対策としてこれまで，

(i) 訪れる順番に並べた n 個の都市を表現する遺伝子列（染色体）を n 個の文字の順列と考え，順列に対する互換や表現方法に対して工夫をこらすことで致死遺伝子が生じないような遺伝子交叉を実現した Grefenstette 法[5]，PMX（Partially Matched Crossover）法[4]，CX（Cycle Crossover）法[6]，OX（Order Crossover）法[7]，(ii) 隣接都市間の所要距離等枝のつながりに着目した EX（Edge Recombination Crossover）法[8]，(iii) 両親の有効な枝情報を子供に伝える SXX（Sub-tour Exchange Crossover）法[10]，EXX 法[9]，EAX 法[11],[12] 等の様々な遺伝子交叉オペレータが発見されている。

一方，ACO は都市間の距離の短さと都市間のフェロモン量の多いさから次に訪れる都市を確率的に選択する手法である。フェロモン量は蟻が都市一周旅行で学習した「巡回路長」の逆数相当の情報で，その巡回路を構成する各都市間の道路上に記憶される。

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（$r>=4$）が可能である。シミュレーテッドアニーリング手法（SA）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。

これまでの我々の C プログラミング実験の結果，(i) 比較的初期の世代では，ある都市からある都市までの部分経路が最適となっている経路を生成するのに，所要距離の最も短い都市を親の隣接都市リストの中から次々に選択して次の訪問都市を決定する改良 EX 法や，都市間の距離の短さに加えて都市間のフェロモン量から確率的に選択する ACO 法が有効であること，(ii) 世代が経るにつれ，上記の部分的に最適となっている部分枝と部分枝を選択し遺伝子交叉させて広域的に最適な経路を生成するのに，親の部分枝をランダムに選択入れ替えて新たな経路を生成する SXX 法，EXX 法や EAX 法が有効であることがわかっている[22],[23],[24],[25]。

このため所要距離の短い経路をより効率的に選択可能とする手法として，遺伝子交叉オペレータとその交替時期を柔軟に選択可能な拡張遺伝子交叉オペレータ交代法 ECXO について研究することとした。本検討では ACO，SA，Lin-Kernighan を選択，交叉や突然変異といった遺伝子操作機能を有する GAs の拡張とみなし，この 3 つのオペレータを拡張遺伝子交叉オペレータと呼ぶ。ECXO の制御対象は GAs の 8 つの遺伝子交叉オペレータ Grefenstette，PMX，CX，OX，EX，SXX，EXX，EAX と ACO，SA，Lin-Kernighan の 3 つの拡張遺伝子交叉オペレータを合わせた合計 11 個の遺伝子交叉オペレータである。ECXO はこれら 11 個の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はそれを任意の時点で参照引き継ぎ更新可能とする。ECXO は同一 Chromosomes を削除する機能や巡回路長の昇順や降順に Chromosomes を並べ直す等の運用支援機能を有する。

3.2 ECXO で制御対象となる拡張遺伝子交叉オペレータの特徴

本実験での ECXO の制御対象となった遺伝子交叉オペレータは EAX，EXX，EX，ACO でありその機能を C 言語[19],[20] で開発した。GAs における個体や ACO における蟻の旅が TSP の巡回路を表現する。我々が実現した各遺伝子交叉オペレータの特徴を以下に整理する。

(1) 枝組み立て交叉（EAX：Edge Assembly Crossover）

本研究での EAX は巡回路を都市と都市を結ぶ有向枝の集合として捉える。EAX は親 A と親 B の同数の枝を交換して子供を次のように生成する。先ず，親 A の枝を順方向に親 B の枝

を逆方向に交互に辿って「A-B サイクル」と呼ばれる幾つかの巡回路を構成する。ここで A-B サイクルを構成する A の枝の終点と B の枝の終点ならびに，A の枝の始点と B の枝の始点は一致している。次に A-B サイクルを構成する枝を相互に交換して局所的に最適な「緩和個体」と呼ばれる部分巡回路群を親 A と親 B から構成する。最後に 2 つの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新たな部分巡回路を生成するという処理を繰り返して大域的に最適な一つの巡回路を再構成する。そのような方法で EAX は両親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。本研究の EAX は以下の特徴を有する。

①　A-B サイクルの構成法

巡回路をグラフで表現する。論文[11]では STSP（対称 TSP）用の無向グラフと ATSP（非対称 TSP）用の有向グラフ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。本研究では無向グラフに適当な向きをつけ有向グラフに変換してから A-B サイクルを構成した。

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の選択の仕方により，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選択する。② で述べたように，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。更にその子供を両親として孫を EAX で 2 個体生成する。この方法で Ncross 回 EAX による遺伝子交叉を繰り返して子供を生成する。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を中止する。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 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である。

(2)　アーントコロニー最適化手法（ACO : Ant Colony Optimization）

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。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。フェロモン量は，蟻が旅行で学習した「巡回路長」の逆数相当の情報であり，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。本検討での ACO は，局所最適解に陥らないように，探索経路を 2-opt 法と 3-opt 法を適用して変更・改善する。変更・改善の際，距離の短縮度合いとこれまでの経過時間を考慮して確率的に判断するシミュレーテッドアニ―リング機能を具備している[25]。

(3)　EX（Edge Recombination Crossover）

隣接都市の中で距離の最も短い都市を次々に辿る最近近傍アルゴリズムの応用である。各々の都市について，親 CX と親 CY の閉路上で隣接する都市の和集合を考え，それを各都市の隣接リストと呼ぶ。第 1 番目の子の最初の訪問都市は親 CX の最初の訪問都市とし，その訪問先の隣接リストの中からまだ訪問していない都市の中で最も距離の短い都市を二番目に訪れる都市とする。こうして隣接リストから次に訪れる都市を次々に選択する。隣接リスト中に新たに訪れる都市がなく，まだ訪問先が残っている場合は未訪問先の中で最も距離の短い都市を次の訪問先として選ぶ。次に訪問する都市がなくな

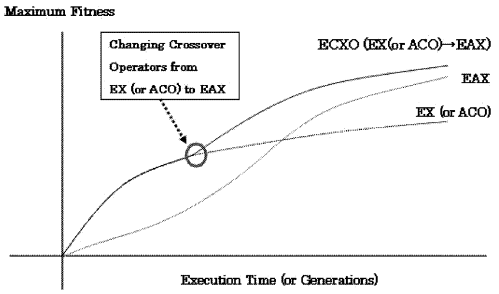

Fig. 1. Concept of performance improvement to solve TSP by ECXO (EX (or ACO)→ EAX)

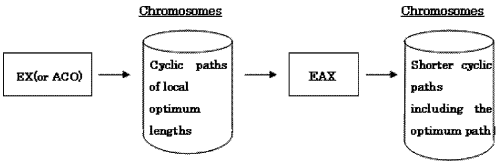

Fig. 2. In ECXO (EX (or ACO)→ EAX) , EX(or ACO) delivers chromosomes generated by himself to EAX.

るまでこの処理を続けて，第 1 番目の子の巡回路を決定する。第 2 番目の子の最初の訪問都市を親 CY の最初の訪問都市とすることから始めて，同様な手順で第 2 番目の子の巡回路を決定する。

(4) EXX（Edge Exchange Crossover）

EXX では親の枝情報を全て次の世代の子供の枝情報として伝える。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。EXX は EAX に比較しロジックが単純で子の生成効率も高いが，両親から生成される子のバリエーションが少ないため都市数が多くなり問題が複雑になると局所最適解に陥り易い。

3.3 拡張遺伝子交叉オペレータ交代法 ECXO（EX(or ACO)→ EAX）

主遺伝子交叉オペレータを EAX，副遺伝子交叉オペレータを EX または ACO とする拡張遺伝子交叉オペレータ ECXO(EX (or ACO) → EAX)の概念を図 1 に示す。両親が部分的に巡回路の長さが最短ならばそれを結合して大域的に最短な解を生成する能力に関しては，EAX は EX または ACO より優れている。しかし，枝の長さに関する情報から子供を生成する機能がないため，EAX 単独では局所的に最適な解を生成するのに時間がかかる。一方副遺伝子交叉オペレータ EX または ACO は両親の枝の長さに関する情報から子供の経路を決定するため局所的に最適な解を生成する能力は EAX より優れているが，大域的な枝と枝のつながりに関する情報から子供を生成できないため局所最適解に陥りやすい。このため，本検討の ECXO では比較的早期の世代では遺伝子交叉オペレータ EX または ACO により局所的に最適な経路を生成し，後期の世代では EAX により大域的に最適な解を生成する。ECXO は EX または ACO が局所的に最適な解を生成した後で，かつそれによる解の収束が始まり多様性が確保できなくなる前に EX または ACO から EAX に交代させる。図 1 の○印の時点で EX または ACO から EAX に遺伝子交叉オペレータを交代させる。図 2 に，ECXO(EX (or ACO) → EAX)の実現方式を示す。図 2 に示すように，EAX は EX または ACO が生成した解を Chromosomes というファイルで引き継ぐ。ECXO は，副遺伝子交叉オペレータ EX または ACO から主遺伝子交叉オペレータ EAX に交代させる時期を合理的に選択する。実験では，EX が単位時間内で生成する個体数が ACO より大きいことを考慮して，ECXO(EX → EAX)の場合は 5 世代単位，ECXO（ACO → EAX）の場合は 1 世代単位で最適な交代時期を探索している。

3.4 拡張遺伝子交叉オペレータ交代法 ECXO(EX(or ACO)→ EXX → EAX)

拡張遺伝子交叉オペレータ ECXO （EX（or ACO)→ EXX → EAX)は初期世代，中期世代，

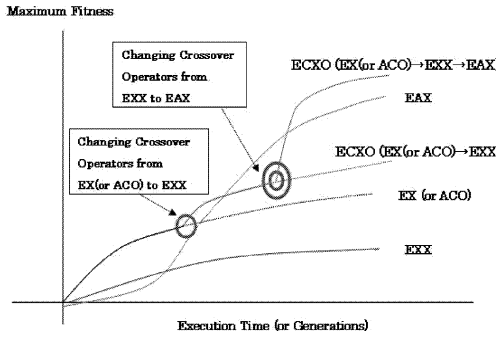

Fig. 3. Concept of performance improvement to solve TSP by ECXO (EX (or ACO)→ EXX → EAX)

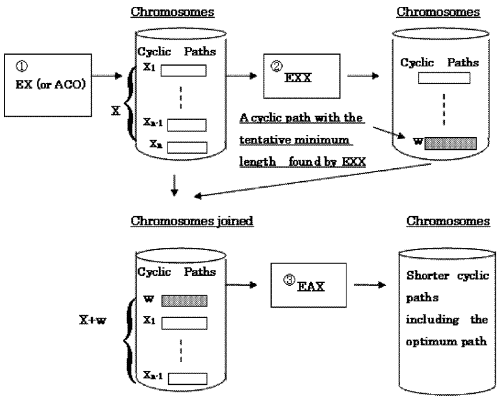

Fig. 4. In ECXO (EX (or ACO)→ EXX → EAX), EX (or ACO) combines EXX to deliver chromosomes generated by themselves to EXX.

後期世代の三段階で遺伝子交叉オペレータを交代させる。その概念を図 3 に示す。その主遺伝子交叉オペレータは遺伝子交叉オペレータ EAX であり，個体の多様性を確保し最適解への収束効率を向上させるために，遺伝子交叉オペレータ EX または ACO の遺伝子操作結果と EXX の遺伝子操作結果を二段階に分けて引き継ぐ。先ず図 3 の○印の時点で EX または ACO から EXX に交代し EXX で解 w への仮収束を図る。次に図 3 の◎印の時点で EXX から EAX に交代し大域的な解を探索する。図 4 は ECXO (EX(or ACO)→ EXX → EAX) の実現方式である。先ず図 4 ① で，副遺伝子交叉オペレータ EX または ACO は部分的に最適だが全体として経路長にばらつきのある多様な個体群 X を生成する。次に図 4 ② で，この遺伝子群 X を初期値として遺伝子交叉オペレータ EXX により解 w への仮収束を図る。遺伝子交叉オペレータ EXX は局所最適解には陥りやすいものの EAX に比べて解の収束効率が良い。そして図 4 ③ で遺伝子交叉オペレータ EX または ACO で生成した遺伝子群 X に遺伝子交叉オペレータ EXX の仮収束解 w を併合させ遺伝子群 $X' = X \vee w$ を生成した後 EAX の初期値として X' を使用し広域的に最適な解をより効率的に探索する。こうして，主遺伝子交叉オペレータ EAX の解の多様性と最適解への収束効率の向上を，経路長にばらつきのある遺伝子群 X と X より巡回路長の短い個体 w を混在させることで実現している。

4. 実験結果

4.1 実験データ

論文 (2)，(12) で参照されている TSPLIB95[26] の 442 都市問題 pcb442 を用いて，EAX を主遺伝子交叉オペレータとする EAX の有効性を検証した。TSPLIB によれば，これまでに見つかっている pcb442 の最短経路長は 50,778 であり，我々の実験でも ECXO と EAX がこれと同じ最短経路を探索した。図 5 に我々が探索した pcb442 の最短経路を示す。

4.2 評価対象の遺伝子交叉オペレータ

以下の 8 つ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 EAX ② EX ③ EXX ④ ACO
⑤ ECXO (EX → EAX)
⑥ EXO (ACO → EAX)
⑦ ECXO (EX → EXX → EAX)
⑧ EXO (ACO → EXX → EAX)

4.3 評価項目

4.2 の遺伝子交叉オペレータを以下の観点で評価した。

① 「探索した最短経路長 L」

② 「最短経路長を探索するまでに要した個体数 N」

最適解を探索するのに必要な生成個体(経路)数 N は EX, EXX では「人口数×最短経路長を探索するまでに要した世代数×2」，EAX では「人口数×最短経路長を探索するまでに要した世代数×2×Ncross」，ACO では「人口数×最短経路長を探索するまでに要した世代数」で計算する。GAs においては，両親が子供を 2 人生成し，適応度の高い子供を残す方式としているため生成個体は人口数の 2 倍とした。Ncross は両親から生成される子孫の最大世代数であり収束が開始すると実効上 1 に近づく。

③ 「最短経路を探索したコンピュータ時間（秒）T」

4.4 プログラム起動パラメータ

本実験での GAs と ACO の起動パラメータは以下の通りである。尚，二都市間の距離は小数点以下四捨五入した整数で求めている。

(1) GAs の起動パラメータ

(A) EAX, EX, EXX 共通の GAs 起動パラメータ

① 乱数種：主に初期集団を決定する。：1

② 人口数：442

③ 遺伝子交叉確率：0.8

④ 最大観測世代数：30(EAX)，200(EX)，5,000(EXX)

⑤ 親の選択方式：ルーレット方式（EX, EXX のみ)

⑥ 世代交代方式：次世代の子が「人口数」分生成されたタイミングで生成した子を次世代の親として一括世代交代する。

⑦ 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

⑧ 2-opt 法を適用した後に 3opt 法を 1 回適用して距離の改善を図る。：YES

⑨ 親と子供を併せた個体群の間でトーナメントを行い適応度の高い個体を残す。：YES

⑩ 生成した 2 人の子供のうち適応度の高い子供のみ残す。：YES

(B) EAX 固有の GAs 起動パラメータ

① 親 A として現世代の個体全てを対象として選ぶ。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：1

② 3.2(1)③ で述べた「エリート戦略」における Ncross 回数：100

(2) ACO の実行パラメータ[25]

(a) 数式モデル

① 都市 i に居る蟻 A_k が都市 j を訪れる確率 $p_{ij}^k(t+1)$

時間＝$t+1$ に都市 i にいる蟻 A_k は次に訪

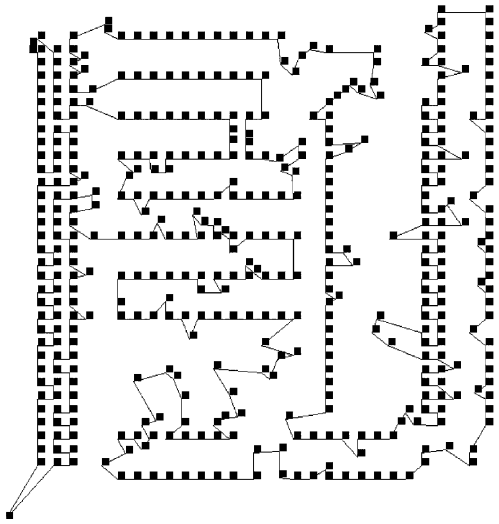

Fig. 5. Test data of pcb442 and its optimum solution with the minimum length of 50,778.

れる都市 j を，都市 i と都市 j を結ぶ枝上に残された t 時におけるフェロモン量 $\tau_{ij}(t)$ と，都市 i と都市 j 間の距離 d_{ij} から決定される以下の確率 $p_{ij}^{k}(t+1)$ により選択する。ここで蟻 A_k はこれまで q 都市のうち l 個の都市について訪問が終わっているものとし，$q-l$ 個の未訪問の都市から次に訪れる都市を決定する。

$$p_{ij}^{k}(t+1)=\frac{\tau_{ij}(t)/d_{ij}^{\beta}}{\sum_{j=1}^{q-l}\tau_{ij}(t)/d_{ij}^{\beta}} \cdots\cdots\cdots \text{式 1}$$

②　フェロモン量更新 $\Delta\tau_{ij}(t+1)$

フェロモン $\tau_{ij}(t+1)$ 量は，現在のフェロモン量 $\tau_{ij}(t)$ と $t=t+1$ の旅で更新したフェロモン更新量 $\Delta\tau_{ij}(t+1)$ から，m 匹の蟻が旅を完了する度に，以下の式で更新される。時間 t は m 匹の蟻が旅を完了する度に更新される。

$$\Delta\tau_{ij}(t+1)=\sum_{k=1}^{m}Q_{ij}(k),$$
$$\tau_{ij}(t+1)=(1-\rho)^{*}\tau_{ij}(t)+\rho^{*}\Delta\tau_{ij}(t+1)$$
$$\cdots\cdots\cdots \text{式 2}$$

ここで，ρ はフェロモンの忘却度である。$Q_{ij}(k)$ は蟻 k が i から j の経路を選択する大きさを測る尺度であり，「蟻 が，都市 i から j を経由する巡回路を旅した場合は，巡回路長の逆数＊二都市間の距離の総和（すなわち GAs での適合度であり，都市 i から j を経由しない場合は 0」である。

(b)　実行パラメータ

①　乱数種：主に蟻の初期旅行経路と経路長を決定する。：1

②　次に訪問する都市を隣接する cl 個の都市の中から優先的に次に訪れる都市を選択する際の cl を指定する。：20

③　人口数 p。ACO は p 匹の蟻の旅行経路と経路長に関するデータを GAs に引き継ぐ。p は蟻が初めて訪れる異なる都市の数である。：442

④　観測最大世代数：200

⑤　都市 i と都市 j 間の辺上に蓄積されるフェロモン量の辺の長さによる重みづけ β（式 1）：2

⑥　フェロモン忘却率 ρ（式 2）：0.3

⑦　フェロモン更新タイミング m；何匹の蟻の旅度にフェロモンを更新するか：442

⑧　蟻の探索経路に 2-opt 法と 3-opt 法を 1 回ずつ適用して距離の改良を図る SA（シミュレーテッド・アニーリング）を利用するか否か。本 SA では，距離の短縮度合いとこれまでの経過時間を考慮しながら，2-opt 法と 3-opt 法を適用して蟻の探索経路を確率的に変更・改善するボルツマンマシンを実現する。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。：YES

4.5　実験結果

(1)　「探索した最短経路長 L」，「最短経路を探索したコンピュータ時間（秒）T」，「最短経路長を探索するまでに要した個体数 N」の評価

4.2 で述べた 8 つの遺伝子交叉オペレータについて pcb442 の例題データを用いて L, T, N を評価した結果を表 1 に示す。表 1 のコンピュータ時間は，NEC Versa Pro Mobile Intel（R）CPU, 2.19 GHz, 240 MB RAM で計測した。表 1 にて○と◎は最短経路長 50,778 を探索したモデル，◎はその中で最も少ないコンピュータ時間で最短経路長を探索したモデルである。参考として 2-opt, 3-opt をベースとした Lin-Kernighan，SA による計測結果も表 1 に添付した。その場合，初期値はランダムであり N は有効状態遷移回数である。

以下に結果をまとめる。

(i)　EAX は 22 世代目 11,393 秒で最短経路長は 50,778 を探索していること，

(ii)　EX 単独では経路長 56,467 の解しか探

Table 1. Comparison of ECXOs with another Crossover Operators EAX, EX, EXX, and ACO on the problem pcb442. (The optimum length of pcb442 found is 50,778)

NO	method	best length found	computer time to find the best length (seconds)	number of tours generated to find the best length	generation to change crossover operators	generation to find the best solution	population size	Ncross
○1	EAX	50,778	11,393	2,210,000		25	442	100
2	EX	56,467	1,304	173,264		196	442	
3	EXX	55,532	1,104	4,294,472		4,858	442	
4	ACO	55,480	20,260	64,532		146	442	
○5	ECXO(EX->EAX)	50,778	8,093	1,599,156	9	27	442	100
○6	ECXO(ACO->EAX)	50,778	9,841	1,857,726	3	24	442	100
◎7	ECXO(EX->EXX->EAX)	50,778	7,861	3,566,940	9; 2426	2,451	442	100
○8	ECXO(ACO->EXX->EAX)	50,778	10,153	3,221,296	4; 1542	1,567	442	100
9	Lin-Kernighan(2opt)	54,211	13,520	2,191		2,191	1	
10	Lin-Kernighan(3opt)	53,910	9,844	2,200		2,200	1	
11	Simulated Annealing	493,940	34,749	1,362		1,362	1	

○ the model which finds the optimum length ◎ the optimum model which has the least computer execution time among all the models

索できないこと，EXX 単独では経路長 55,532 の解しか探索できないこと，ACO 単独では経路長 55,480 の解しか探索できないこと，初期値がランダムなため Lin-Kernighan （2-opt，3-opt），SA 単独では経路長 54,211，53,910，493,940 の解しか探索できないこと，

(iii) EX から 9 世代目に EAX に交代する ECXO（これを ECXO(EX → EAX) と記す）は 27 世代目 8,093 秒で最短経路長 50,778 を探索していること，

(iv) ACO から 3 世代目に EAX に交代する ECXO(ACO → EAX) は 24 世代目 9,841 秒で最短経路長 50,778 を探索したこと，

(v) EX から 9 世代目に EXX に交代し，EXX から EAX に 2,426 世代目に交代する ECXO (EX → EXX → EAX) は 2,451 世代目 7,861 秒で最短経路長 50,778 を探索していること，ここで ECXO (EX → EXX → EAX) では解の多様性を確保しかつ効率的な最適解の探索を可能とするため EX 法で探索した 441 個の解に EXX 法で探索した最適解 1 個を混在させた 442 個の遺伝子群を EAX の入力としている。

(vi) ACO から 4 世代目に EXX に交代し，EXX から EAX に 1,542 世代目に交代する ECXO (ACO → EXX → EAX) は 1,567 世代目 10,153 秒で最短経路長 50,778 を探索していることを示している。ここで (v) と同様に ACO で探索した 441 個の解に EXX 法で探索した最適解 1 個を混在させた 442 個の遺伝子群を EAX の入力としている。

上記のことから，ECXO (EX → EAX)，ECXO(ACO → EAX)，ECXO(EX → EXX → EAX)，ECXO(EX → EXX → EAX) の各遺伝子交叉オペレータ交代法により EAX の最適解探索時間を 10%～30% 削減できること，本事例では ECXO (EX → EXX → EAX) が探索時間を最も削減できることがわかった。

(2) ECXO における遺伝子交叉オペレータ交代時期の選択

表 1-1～表 1-4 にて◎と○は表 1 のそれと同じ役割を持つ。

① ECXO(EX → EAX) における EX から EAX への交代世代の選択（表 1-1 参照）

5 世代目，9 世代目，15 世代目に EX から EAX に遺伝子交叉オペレータを交代させる ECXO(EX → EAX) について調査した結果，全てのモデルで EAX が最適解 50,778 を探索するのに要した 11,393 秒より少ないコンピュータ時間で同じ最適解 50,778 を探索できた。そのうち 9 世代に EX から EAX に交代させる ECXO(EX → EAX) は最も少ないコンピュー

Table 1-1.　Selection of generation to change crossover operators from EX to EAX

NO	method	best length found	computer time to find the best length (seconds)	number of tours generated to find the best length	generation to change crossover operators	generation to find the best solution	population size	Ncross
○5-1	ECXO(EX->EAX)	50,778	10,341	2,037,620		28	442	
○5-1-1	EX	85,807	50	4,420	5	5	442	
○5-1-2	EAX	50,778	10,291	2,033,200	-	23	442	100
◎5-2	ECXO(EX->EAX)	50,778	8,093	1,599,156		27	442	
◎5-2-1	EX	61,801	86	7,956	9	9	442	
◎5-2-2	EAX	50,778	8,007	1,591,200	-	18	442	100
○5-3	ECXO(EX->EAX)	50,778	9,116	1,781,260		35	442	
○5-3-1	EX	59,785	136	13,260	15	15	442	
○5-3-2	EAX	50,778	8,980	1,768,000	-	20	442	100

○ the model which finds the optimum length　◎ the optimum model with the least computer execution time in ECXO (EX → EAX)

Table 1-2.　Selection of generation to change crossover operators from ACO to EAX

NO	method	the best length found	computer time to find the best length (seconds)	number of tours generated to find the best length	generation to change crossover operators	generation to find the best solution	population size	Ncross
○6-1	ECXO(ACO->EAX)	50,778	10,483	2,034,084		25	442	
○6-1-1	ACO	80,162	266	884	2	2	442	
○6-1-2	EAX	50,778	10,217	2,033,200	-	23	442	100
◎6-2	ECXO(ACO->EAX)	50,778	9,841	1,857,726		24	442	
◎6-2-1	ACO	73,650	399	1,326	3	3	442	
◎6-2-2	EAX	50,778	9,442	1,856,400	-	21	442	100
○6-3	ECXO(ACO->EAX)	50,778	10,390	1,946,568		26	442	
○6-3-1	ACO	69,817	531	1,768	4	4	442	
○6-3-2	EAX	50,778	9,859	1,944,800	-	22	442	100

○ the model which finds the optimum length　◎ the optimum model with the least computer execution time in ECXO (ACO → EAX)

Table 1-3.　Selection of generation to change crossover operators from EX to EXX before operating EAX

NO	method	the best length found	computer time to find the best length (seconds)	number of tours generated to find the best length	generation to change crossover operators	generation to find the best solution	population size	Ncross
○7-1	ECXO(EX->EXX->EAX)	50,778	12,018	5,989,100		4,300	442	
○7-1-1	EX	85,807	50	4,420	5	5	442	
○7-1-2	EXX	56,007	695	3,774,680	4,270	4,270	442	
○7-1-3	EAX	50,778	11,273	2,210,000	-	25	442	100
◎7-2	ECXO(EX->EXX->EAX)	50,778	7,861	3,566,940		2,451	442	
◎7-2-1	EX	61,801	86	7,956	9	9	442	
◎7-2-2	EXX	53,384	387	2,144,584	2,426	2,426	442	
◎7-2-3	EAX	50,778	7,388	1,414,400	-	16	442	100
○7-3	ECXO(EX->EXX->EAX)	50,778	10,021	6,073,080		4,791	442	
○7-3-1	EX	59,785	136	13,260	15	15	442	
○7-3-2	EXX	53,117	737	4,203,420	4,755	4,755	442	
○7-3-3	EAX	50,778	9,148	1,856,400	-	21	442	100

○ the model which finds the optimum length　◎ the optimum model with the least computer execution time in ECXO (EX → EXX → EAX)

タ時間 8,093 秒で最適解を探索できた。表 1 で示した ECXO(EX → EAX) は 9 世代に EX から EAX に交代させるモデルの ECXO(EX → EAX）である。

Table 1-4. Selection of generation to change crossover operators from ACO to EXX before operating EAX

NO	method	the best length found	computer time to find the best length (seconds)	number of tours generated to find the best length	generation to change crossover operators	generation to find the best solution	population size	Ncross
○8-1	ECXO(ACO->EXX->EAX)	50,778	10,484	5,351,736		3,877	442	
○8-1-1	ACO	80,162	266	884	2	2	442	
○8-1-2	EXX	54,215	679	3,406,052	3,853	3,853	442	
○8-1-3	EAX	50,778	9,539	1,944,800	-	22	442	100
○8-2	ECXO(ACO->EXX->EAX)	50,778	11,245	3,819,322		1,946	442	
○8-2-1	ACO	73,650	399	1,326	3	3	442	
○8-2-2	EXX	55,532	323	1,696,396	1,919	1,919	442	
○8-2-3	EAX	50,778	10,523	2,121,600	-	24	442	100
◎8-3	ECXO(ACO->EXX->EAX)	50,778	10,153	3,221,296		1,567	442	
◎8-3-1	ACO	69,817	531	1,768	4	4	442	
◎8-3-2	EXX	54,115	259	1,363,128	1,542	1,542	442	
◎8-3-3	EAX	50,778	9,363	1,856,400	-	21	442	100

○ the model which finds the optimum length ◎ the optimum model with the least computer execution time in ECXO (ACO → EXX → EAX)

Table 2. Comparison of ECXOs and other crossover operators on execution time to find their best lengths for pcb442.

gen	execution time (sec)	best length							
		EAX	EX	EXX	ACO	ECXO(EX→EAX)	ECXO(ACO→EAX)	ECXO(EX→EXX→EAX)	ECXO(ACO→EXX→EAX)
0	10	729,120	729,120	729,120	730,254	729,120	730,254	480,190	730,254
1	562	67,597	57,718	89,891	69,187	61,801	73,650	53,384	69,187
2	1,192	55,839	56,633	55,532	67,335	54,440	54,972	53,279	54,115
3	1,740	55,688	56,467	55,532	64,422	53,607	54,372	53,015	53,453
4	2,266	55,346	56,467	55,532	64,422	53,435	54,134	52,754	53,184
5	2,837	54,153	56,467	55,532	63,375	52,530	53,578	52,533	52,624
6	3,359	53,953	56,467	55,532	62,765	52,456	53,042	52,366	52,394
7	3,900	53,529	56,467	55,532	62,372	51,953	52,465	52,070	52,153
8	4,376	53,421	56,467	55,532	62,372	51,560	52,316	51,876	52,040
9	4,883	52,823	56,467	55,532	62,372	51,430	51,958	51,535	51,746
10	5,342	52,153	56,467	55,532	61,947	51,381	51,788	51,359	51,639
11	5,798	52,004	56,467	55,532	61,947	51,189	51,613	51,164	51,449
12	6,245	51,780	56,467	55,532	61,679	51,104	51,493	50,961	51,230
13	6,700	51,621	56,467	55,532	61,679	50,876	51,330	50,847	51,191
14	7,175	51,332	56,467	55,532	61,679	50,864	51,206	50,811	51,135
15	7,668	51,234	56,467	55,532	61,679	50,818	50,895	50,785	51,001
16	8,055	51,175	56,467	55,532	61,293	50,792	50,895	50,778	51,001
17	8,468	51,013	56,467	55,532	61,293	50,778	50,852	50,778	50,890
18	8,842	50,925	56,467	55,532	60,696	50,778	50,841	50,778	50,844
19	9,215	50,861	56,467	55,532	60,696	50,778	50,818	50,778	50,821
20	9,612	50,833	56,467	55,532	60,696	50,778	50,799	50,778	50,792
21	9,982	50,808	56,467	55,532	60,696	50,778	50,778	50,778	50,785
22	10,375	50,786	56,467	55,532	59,885	50,778	50,778	50,778	50,778
23	10,709	50,785	56,467	55,532	59,885	50,778	50,778	50,778	50,778
24	11,063	50,780	56,467	55,532	57,975	50,778	50,778	50,778	50,778
25	11,393	50,778	56,467	55,532	57,975	50,778	50,778	50,778	50,778

② ECXO(ACO → EAX) における ACO から EAX への交代世代（表 1-2 参照）

2 世代目，3 世代目，4 世代目に ACO から EAX に交代させる ECXO(ACO → EAX) について調査した結果，いずれも EAX が最適解 50,778 を探索するのに要した 11,393 秒より少ないコンピュータ時間で同じ最適解を探索できていること，その中で 3 世代に ACO から EAX に交代させる ECXO (ACO → EAX) は最も少ないコンピュータ時間 9,841 秒で最適解を探索できることがわかった。表 1 で示した ECXO (ACO → EAX) は 3 世代に ACO から EAX に交代させるモデルの ECXO(ACO → EAX) である。

③ ECXO (EX → EXX → EAX) における EX から EXX への交代世代（表 1-3 参照）

5 世代目，9 世代目，15 世代目に EX から

EXX に交代させ EXX で仮収束を図り，その後 EAX で最終的な解の収束を図る ECXO(EX → EXX → EAX)について調査した結果，(i) 3 つのモデルはいずれも最適解 50,778 を探索していること，(ii) 9 世代目ならびに 15 世代で EX から EXX に交代させ EXX で仮収束を図ったモデルが EAX より少ないコンピュータ時間で同じ最適解を探索できていること，(iii) 9 世代目で EX から EXX に交代させ EXX で仮収束を図ったモデルが最も少ないコンピュータ時間 7,861 秒で最適解を探索できることがわかった。表 1 で示した ECXO(EX → EXX → EAX) は 9 世代に EX から EXX に交代しその後 EXX から EAX に交代する第 2 のモデルの ECXO (EX → EXX → EAX) である。

④ ECXO (ACO → EXX → EAX) における ACO から EXX への交代世代 (表 1-4 参照)

2 世代目，3 世代目，4 世代目に ACO から EXX に交代させ EXX で仮収束を図り，その後 EAX で最終的な解の収束を図る ECXO(ACO → EXX → EAX)について調査した結果，(i) 3 つの ECXO(ACO → EXX → EAX) はいずれも EAX が最適解 50,778 を探索するのに要した 11,393 秒より少ないコンピュータ時間で同じ最適解を探索できていること，(ii) そのうち 4 世代目に ACO から EXX に交代させ EXX で仮収束を図るモデルが最も少ないコンピュータ時間 10,153 秒で最適解を探索できることがわかった。表 1 で示した ECXO(ACO → EXX → EAX) は 4 世代目に ACO から EXX に交代させ EXX で仮収束を図るモデルの ECXO (ACO → EXX → EAX) である。

(3)　最適解探索のプロセス

表 2 は EAX の各世代の終了時刻で EAX ならびに他の 7 つの遺伝子交叉オペレータが探索した最短経路長がどのように変化しているかを示している。それをグラフ化し図 6 に示す。グラフの X 軸は実行時間，Y 軸はその時刻までに探索した最短経路長を示す。表 2，図 6 から，(i) 2 世代目完了時刻 562 sec では EAX が探索した最短 67,597 より短い経路長 57,718 を EX は探索しているが，4 世代目完了時刻 1,740 sec 以降，EX の解は局所最適解 56,467 に陥り解の改善が見られないこと，(ii) 3 世代目完了時刻 1,192 sec 以降，EXX は局所最適解 55,532 に陥り解の改善が見られないこと，(iii) EAX が最適解 50,778 を探索した 25 世代目完了時刻 11,393 sec で ACO は 57,975 の局所最適解した探索できていないことがわかる。図 7 は 14 世代目以降を詳細化したグラフであり，EAX ならびに 4 つの ECXO の最適解 50,778 への収束速度を比較している。表 1，表 2，図 7 から 7,861 sec で最初に ECXO (EX → EXX → EAX) が最適解を探索し，次に ECXO (EX → EAX)，ECXO (ACO → EAX)，ECXO (ACO → EXX → EAX) の順番で最適解を探索し，最後に 11,393 秒で EAX が単独で最適解を探索してい

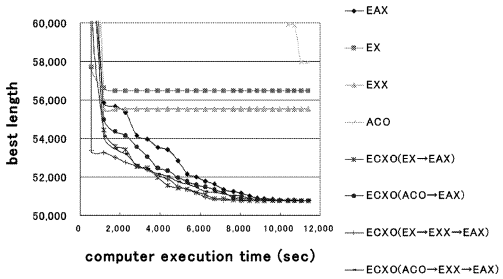

Fig. 6.　Evaluation of both of ECXOs and other crossover operators on computer time to find their best lengths of pcb442.

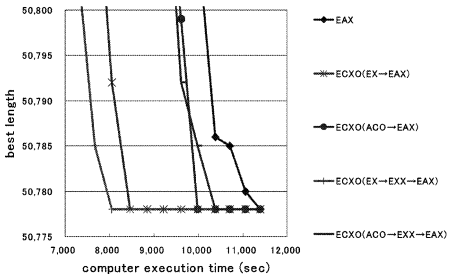

Fig. 7.　ECXOs converge to the optimum solution earlier than EAX on the problem pcb442.

ることがわかる。

5. 他の TSP 問題による ECXO の有効性の検証

pcb442 に加え論文 (2)，(12) で参照されている lin318, d198 により ECXO の有効性を検証した。結果を以下に整理する。

(1) lin318 による ECXO の有効性の検証 (表 3)

表 3 に lin318 を用いた L, N, T に関する 8 つの遺伝子オペレータの比較評価結果を整理する。表 3 のコンピュータ時間は，COMPAC NX6320 Genuine Intel(R) CPU, 1.66 GHz, 504 MB RAM で計測した。表 3 にて○と◎は表 1 のそれと同様な役割を果たしている。結果を以下にまとめる。

(i) EAX は 22 世代目 2,321 秒で最短経路長 42,029 を探索していること，

(ii) EX 単独では経路長 44,300 の解しか探索できないこと，EXX 単独では経路長 44,539 の解しか探索できないこと，ACO 単独では経路長 53,605 の解しか探索できないこと，

(iii) EX から 15 世代目に EAX に交代する ECXO (これを ECXO(EX → EAX) と記す) は 34 世代目 2,291 秒で最短経路長 42,029 を探索していること，

(iv) ACO から 3 世代目に EAX に交代する ECXO(ACO → EAX) は 26 世代目 2,517 秒で最短経路長 42,029 を探索していること，

Table 3. Comparison of ECXOs with another Crossover Operators EAX, EX, EXX, and ACO on the problem lin318. (The optimum length of lin318 found is 42,029)

NO	method	best length found	computer time to find the best length (seconds)	number of tours generated to find the best length	generation to change crossover operators	generation to find the best solution	population size	Ncross
○1	EAX	42,029	2,321	279,840		22	318	20
2	EX	44,300	490	125,292		197	318	
3	EXX	44,539	269	3,080,784		4,844	318	
4	ACO	53,605	32,480	222,600		700	318	
○5	ECXO(EX->EAX)	42,029	2,291	251,220	15	34	318	
○6	ECXO(ACO->EAX)	42,029	2,517	293,514	3	26	318	20
◎7	ECXO(EX->EXX->EAX)	42,029	2,224	1,682,220	15;2250	2,284	318	
○8	ECXO(ACO->EXX->EAX)	42,029	3,055	3,495,774	3;4995	5,023	318	20

○ the model which finds the optimum length ◎ the model which finds the optimum length with least computer execution time

Table 4. Comparison of ECXOs with another Crossover Operators EAX, EX, EXX, and ACO on the problem d198. (The optimum length of d198 found is 15,780)

NO	method	the best length found	computer time to find the best length (seconds)	number of tours generated to find the best length	generation to change crossover operators	generation to find the best solution	population size	Ncross
○1	EAX	15,780	477	110,880		14	198	20
2	EX	16,110	258	76,824		194	198	
3	EXX	16,364	159	1,941,984		4,904	198	
4	ACO	16,570	1,333	16,632		84	198	
◎5	ECXO(EX->EAX)	15,780	321	77,616	23	25	198	
○6	ECXO(ACO->EAX)	15,780	382	87,516	29	13	198	
○7	ECXO(EX->EXX->EAX)	15,780	398	820,908	23;49	1,883	198	
○8	ECXO(ACO->EXX->EAX)	15,780	530	1,959,606	14;128	4,721	198	

○ the model which finds the optimum length ◎ the model which finds the optimum length with least computer execution time

(v)　EX から 15 世代目に EXX に交代し，EXX から EAX に 2,250 世代目に交代する ECXO（EX → EXX → EAX）は 2,284 世代目 2,224 秒で最短経路長 42,029 を探索していること，ここで ECXO（EX → EXX → EAX）は解の多様性を確保するため EX 法で探索した 317 個の解に EXX 法で探索した最適解 1 個を混在させた 318 個の遺伝子群を EAX の入力としている。

(vi)　ACO から 3 世代目に EXX に交代し，EXX から EAX に 4,995 世代目に交代する ECXO（ACO → EXX → EAX）は 5,023 世代目 3,055 秒で最短経路長 42,029 を探索している。ここで ECXO（ACO → EXX → EAX）では ACO で探索した 317 個の解に EXX 法で探索した最適解 1 個を混在させた 318 個の遺伝子群を EAX の入力としている。

表 3 から pcb442 と同様 lin318 においても ECXO（EX → EAX）と ECXO（EX → EXX → EAX）の遺伝子交叉オペレータ交代法は EAX の最適解探索時間を 5% 削減していることが読み取れる。

(2)　d198 による ECXO の有効性の検証（表 4）

表 4 は pcb442，lin318 と同様 d198 においても，遺伝子交叉オペレータ交代法 ECXO と EAX は共に最短経路長 15,780 を探索し，ECXO は EAX の最適解探索時間を 20% ～30% 削減できることを示している。表 4 のコンピュータ時間は，NEC Lavie Intel Pentium III, 695 MHz, 512 MB RAM で計測した。

6. 結　　論

拡張遺伝子交叉オペレータ交代法 ECXO (Extended Changing Crossover Operator) の有効性を TSPLIB の中規模 TSP データ (pcb442, d198, lin318) を用いた C プログラム実験で検証した。当 ECXO は比較的早期の世代では遺伝子交叉オペレータ EX または ACO ならびに EXX により局所的に最適で多様な解を生成し，これを入力として後期の世代では遺伝子交叉オペレータ EAX により大域的に最適な解を効率的に生成する。実験の結果，ACO や EX，EXX 単独では最短経路を探索することが困難なこと，EAX 単独で最短経路を探索できることがわかった。EAX 単独で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。本実験により，ECXO は EAX の最適解探索時間を 5%～30% 向上できること等を検証した。

追記

本論文は以下の国際会議論文の邦訳である。

(27) Ryouei Takahashi, “A Methodology of Extended Changing Crossover Operators to Solve the Traveling Salesman Problem,” The 4th International Conference on Natural Computation (ICNC’08) and The 5th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD ’08), pp. 263–269, IEEE Computer Society, China Jinan, 2008.

参 考 文 献

(1) M. Dorigo and T. Stutzle : “Ant Colony Optimization”, The MIT Press (2004)
(2) M. Dorigo and L.M. Gambardella : “Ant Colony System : A Cooperative Learning Approach to the Traveling Salesman Problem”, IEEE Transactions on Evolutional Computation, 1(1), pp. 53–66 (1997)
(3) J.H. Holland : “Adaptation in Natural and Artificial Systems”, MIT Press (1992)
(4) D.E. Goldberg : “Genetic Algorithms in Search, Optimization, and Machine Learning”, Addison-Wesley Publishing Company, Inc. (1989)
(5) 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)
(6) 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)
(7) L. Davis : "Applying Adaptive Algorithms to Epistatic Domains", Proc. of 9th Int. Joint Conf. on Artificial Intelligence, pp. 162-164 (1985)
(8) 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)
(9) 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)
(10) 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)
(11) Y. Nagata and S. Kobayashi : "The proposal and evaluation of a crossover for Traveling Salesman Problem : Edge Assembly Crossover", JSAI, Vol. 14, No. 5, pp. 848-859, 1999 (in Japanese). 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)
(12) 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)
(13) K. Helsgaun : "An Effective Implementation of the Lin-Kernighan Traveling Salesman Heuristics", European Journal of Operational Research, 126(1), pp. 106-130 (2000)
(14) S. Lin & B.W. Kernighan : "An Effective Heuristic Algorithm for the Traveling-Salesman Problem", Oper. Res. 21, pp. 498-516 (1973)
(15) R.L. Haupt and S.E. Haupt : "Practical Genetic Algorithms", John Wiley & Sons, Inc. (2004)
(16) E. Aarts and J.K. Lenstra : "Local Search in Combinatorial Optimization", Princeton University Press (2003)
(17) M. Garey and D. Johnson : "Computers and Intractability, A Guide to the Theory of NP-Completeness", W.H Freeman and Company (1979)
(18) S.J. Russell and P. Norvig : "Artificial Intelligence—A Modern Approach", Prentice Hall, Upper Saddle River, New Jersey (1995)
(19) B.W. Kernighan and D.M. Richie : "The C Programming Language, Second Edition", Bell Telephone Laboratories (1988)
(20) H. Hirano : "A Genetic Algorithm C programming", Personal Media Inc., pp. 232-238, 1995 (in Japanese). 平野広美：「遺伝的アルゴリズム C プログラミング」，pp. 232-pp. 238, パーソナルメディア社 (1995)
(21) N. Sannomiya, H. Kita, H. Tamaoki, T. Iwamoto : "Genetic Algorithms and Optimization", Asakura-Shoten, 1998 (in Japanese). 三宮信夫，喜多　一，玉置　久，岩本貴司：「遺伝的アルゴリズムと最適化」，pp. 37-51, 朝倉書店 (1998)
(22) R. Takahashi : "Solving the Traveling Salesman Problem through Changing Crossover Operators", Proceedings of the 10th Conference on Artificial Intelligence and Applications, p. 42, hosted by National University of Kaohsiung, 2005.
(23) R. Takahashi : "Solving the Traveling Salesman Problem through Genetic Algorithms with Changing Crossover Operators", Proceedings of Fourth International Conference on Machine Learning and Applications, IEEE Computer Society, pp. 319-324 (2005)
(24) R. Takahashi and K. Degai : "A Performance Improvement of Genetic Algorithms through Changing Crossover Operators to Solve the Traveling Salesman Problem", Proceedings of the 8th International Conference on Computer and Information Technology, organized by Islamic University of Technology, pp. 40-45 (2005)
(25) 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, Xidian University, pp. 114-130 (2006)
(26) TSPLIB95 : http://www.informatik.uni-heidelberg.de/