JAIST Repository
https://dspace.jaist.ac.jp/

Title	局所的な交叉EAXを用いたGAの高速化とTSPへの適用
Author(s)	永田，裕一
Citation	人工知能学会論文誌，22(5)：542-552
Issue Date	2007
Type	Journal Article
Text version	publisher
URL	http://hdl.handle.net/10119/7826
Rights	Copyright (C) 2007 人工知能学会．永田　裕一，人工知能学会論文誌，22(5)，2007，542-552．
Description	

JAIST
JAPAN
ADVANCED INSTITUTE OF
SCIENCE AND TECHNOLOGY

Japan Advanced Institute of Science and Technology

論 文 Technical Papers

局所的な交叉EAXを用いたGAの高速化とTSPへの適用

Fast Implementation of Genetic Algorithm by Localized EAX Crossover for the Traveling Salesman Problem

永田 裕一
Nagata Yuichi
北陸先端科学技術大学院大学 情報科学研究科
Graduate School of Information Science, Japan Advanced Institute of Science and Technology
nagatay@jaist.ac.jp

keywords: genetic algorithm, TSP, EAX, localized crossover, population diversity

Summary

We propose an genetic algorithm (GA) that applies to the traveling salesman problem (TSP). The GA uses edge assembly crossover (EAX), which is known to be effective for solving the TSP. We first propose a fast implementation of a localized EAX where localized edge exchanges are used in the EAX procedure. We also propose a selection model with an effective combination of the localized EAX that can maintain population diversity at negligible computational costs. Edge entropy measure is used to evaluate population diversity. We demonstrate that the proposed GA is comparable to state-of-the-art heuristics for the TSP. Especially, the GA is superior to them on large instances more than 10,000 cities. For example, the GA found an optimal solution of brd14051 (14,051 cities instance) with a reasonable computational cost. The results are quite impressive because the GA does not use Lin-Kernighan local search (LKLS) even though almost all existing state-of-the-art heuristics for the TSP based on LKLS and its variants.

1. はじめに

巡回セールスマン問題（traveling salesman problem, TSP）は最も良く知られたNP困難な組合せ最適化問題の一つである．$G = (V, E, w)$ を N 個の頂点（都市）からなる重み付完全グラフとする．ここで V, E, w はそれぞれ頂点（都市）集合，枝集合，枝のコスト集合である．TSPにおける解（近似解）は，グラフ G 上で最も（なるべく）短い巡回路長を持つハミルトン閉路と定義される．

TSPは問題の定義が簡単で直感的に理解しやすいため，組合せ最適化問題に対して厳密解法や近似解法の新しい枠組みを考案する際のベンチマークとして利用されてきた [久保 94]．それゆえ，TSPに対しては多くの近似解法が提案されており近似解法の洗練度も高い．Johnson & McGeochのサーベイ [Johnson 97, Johnson 02] にあるように，1970年代以降TSPのstate-of-the-art近似解法*1はいずれもLin-Kernighan法（LK法）[Lin 73] と呼ばれる局所探索を改良した手法であり，現状ではLK法を用いることなくこれらの手法の性能を越えることは困難であると考えられている．Iterated Lin-Kernighan (ILK) [Martin 91, Johnson 97] はLK法をIterated local searchの枠組みに拡張した方法であり，それ以降のTSPのstate-of-the-art近似解法の基本型となっている．Chained Lin-Kernighan (CLK) [Cook 00] はILKのアイデアを洗練化した手法で，現在最も効率的なTSPの近似解法の一つと考えられている．また，Heslgaun's LK (LKH) [Helsgaun 00] は，TSPの緩和問題である1-木の考察に基づいて，LK法における探索に効率的な枝狩りを導入した手法である．近年における大規模TSPベンチマークの最良解の多くはLKHを使ったIterated local searchにより発見されている [National TSP].

TSPに対しては遺伝的アルゴリズム（Genetic algorithm, GA）も数多く適用されてきた．GAの探索性能は探索オペレーターである交叉に大きく依存するため，TSPに対しては数多くの交叉が提案されている [Whitley 89, 山村 92, 前川 95]．特に，枝組み立て交叉（Edge Assembly crossover, EAX）[永田 99] は計算コストと計算精度の両面においてTSPの交叉として優れており，EAXがなぜ良い性能を示すのかを解析した研究 [Watson 00, Nagata 04]，EAXを改良した研究 [Ikeda 02, Chan 05, 永田 06, Nagata 06]，EAXを用いたGAとLK法

*1 近似解法の性能は計算コストと近似精度によって決まる．よって，この二つの指標に関するパレート点を実現する近似解法が良い近似解法とされる．本論文では，計算時間を比較的長く許した場合における近似解法を研究（比較）対象とする．

のハイブリッド法である HeSEA [Tsai 04]，などの関連研究がある．しかし，EAX を用いた GA は高い近似精度を実現するものの，CLK，LKH などの LK 法を改良した局所探索法と比較して大くの計算コストを要することが問題であった．近年では，GA と LK 法を組合せたハイブリッド法 [Merz 97, Bonachea 00, Tsai 04] により良い成果が報告されているが，これらの方法では LK 法が探索性能に決定的に重要な役割を果たしている．

本研究の目的は交叉 EAX を用いた GA を改良し，TSP の state-of-the-art 近似解法を超える性能を実現することである．最近 30 年間におけるこれらの近似解法がすべて LK 法に基づいた手法であることから，GA 単体で（LK 法とのハイブリッドを用いずに）これらの手法に匹敵あるいは優れた近似解法を実現することは，TSP に関連した研究分野に大きな影響をもたらすものと言える．また，GA が局所探索に対して多くの計算時間を要することは，TSP における場合に限らず，GA の一般的な問題点の一つである．TSP のような代表的な組合せ最適化問題で，この問題点を指摘し改善することは，組合せ最適化問題一般に対する GA 設計のヒントになるものと期待できる．本論文では以下の 2 つの方法を提案する．

第一に，EAX を高速に実行する実装法を提案する．TSP の交叉オペレーターは，通常，両親となる二つの巡回路間での大域的な（オーダー N の本数の）枝の交換により子個体を生成する．そのため子個体を一つ生成する際の計算量を $O(N)$ より小さくすることは原理的に不可能である．また，従来の実装 [永田 99] でこの計算量を実現している．一方，[永田 00, 永田 06] では枝の交換を局所的に制限した EAX を用いた GA により，精度の良い近似解が得られることを示している．ただし，従来の実装では一個体を生成するための計算量は依然として $O(N)$ である．本論文では局所的な枝の交換を行う EAX を実行する際に，交叉の手続も局所的に行うことで計算量を削減する（交換される枝の数の関数オーダーとする）ためのデータ構造とアルゴリズムを提案する．

第二に，GA 集団の多様性を効率的に維持して GA の探索精度を向上させる方法を提案する．一般に，集団の多様性維持を考慮する場合には何らかの付加的な計算を行う必要があるが，これが GA 全体の計算コストを占めるようでは問題がある．[永田 06] では枝の交換を局所的に制限した EAX を用いた場合，局所的な多様性の損失と呼ばれる指標を計算することで，無視できる程度の計算コストの増加で集団の多様性を維持できることを示した．本論文では同手法を発展させ，多様性の指標に枝エントロピー [前川 97] を用いる方法を提案する．

本論文の以下の構成は，2 章で EAX の概要を述べる．3 章で局所的 EAX の高速実装法，4 章では効率的な多様性維持の方法を提案する．5 章で提案手法の効果を検証し，6 章では他手法との探索性能の比較を行う．7 章で考察を行い，8 章はまとめである．

2. EAX の 概 要

本章では [永田 99, 永田 06] で述べられている EAX の基本的な手続きを示す．

2･1 EAX の手続き

§1 基本ステップ

以下と図 1 に EAX の基本ステップを示す．まず用語を定義する．E_A と E_B をそれぞれ tour-A と tour-B（両親）を構成する枝の集合とする．また，E_C を生成される子個体を構成する枝の集合とする．G_{AB} を $E_A \cup E_B$ で定義される多重グラフとする．ただし，両親で共通する枝は多重枝として扱う．*AB-cycle* を G_{AB} 上で $e_A(\in E_A)$ と $e_B(\in E_B)$ を交互にたどって得られる閉路と定義する．*E-set* を任意の *AB-cycle* の組合せに含まれる枝の集合と定義する．

1 tour-A と tour-B から G_{AB} を構成する．

2 G_{AB} 上の枝を *AB-cycle* へと分割する[*2]．(図では 12 個の *AB-cycle* が示されている)．

3 生成された全ての *AB-cycle* から幾つかを何らかの基準に従い選択し，*E-set* を構成する．ただし，一対の多重枝のみからなる *AB-cycle* は候補から外す（図では 3 種の *E-set* が示されている.）．

4 *E-set* に含まれる tour-A の枝を tour-A から取り除き，これに *E-set* に含まれる tour-B の枝を付け加えることで中間個体を生成する．すなわち，$E_C := E_A$, $E_C := (E_C \backslash (\textit{E-set} \cap E_A)) \cup (\textit{E-set} \cap E_B)$ とする．

5 部分巡回路からなる中間個体をつなぎ合わせ，完全な巡回路へと修正する（詳細は 2･1･2 節に述べる）．

6 さらに子を生成する場合は 3 へ．そうでなければ終了．

§2 修正操作

まず用語の定義を行う．中間個体を構成する枝集合 E_C において，m を中間個体を構成する部分巡回路の個数とする．また，A_l $(l = 1, \ldots, m)$ を l 番目の部分巡回路に含まれる都市の数とする．

5-1 枝集合 E_C より，m と A_l $(l = 1, \ldots, m)$ を得る．

5-2 最小の A_l を持つ部分巡回路の番号を r とする．また，この部分順回路を構成する枝の集合を U とする．

5-3 $\arg \max_{\substack{e \in U \\ e' \in E_C \backslash U}} \{-w(e) - w(e') + w(e'') + w(e''')\}$ を探索し，$\{e^*, e'^*, e''^*, e'''^*\}$ とする．ここで，e'', e''' は Fig. 図 2(a) に示されるように，e と e' で切断された各部分巡回路を結合するように選ばれる．

5-4 枝 e'^* を含む部分巡回路の番号を r' とする．$E_C := (E_C \backslash \{e^*, e'^*\}) \cup \{e''^*, e'''^*\}$ として，r 番目と r' 番目の部分巡回路を結合する．

5-5 $m := m - 1$ として，更新された E_C に対し A_l $(l = 1, \ldots, m)$ を計算．m が 1 なら E_C が巡回路となり終了．そうでなければ 5-2 へ．

*2 G_{AB} に含まれる枝は余すことなく *AB-cycle* へと分割されるが，この分割は一意ではない．

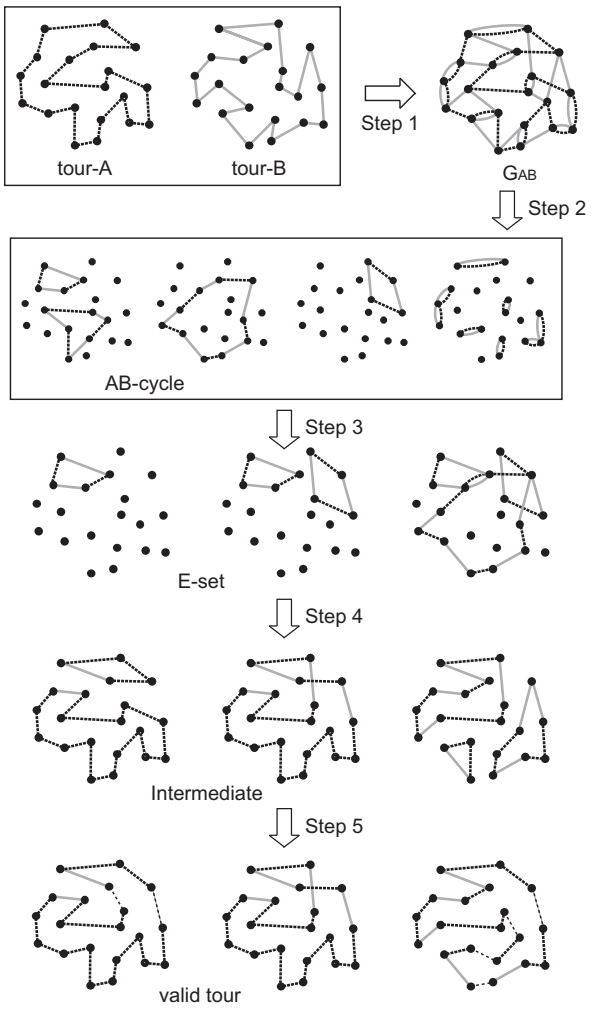

図 1 EAX の基本ステップ

5-1 ではその後のステップのために中間個体を構成する部分順回路の個数と各部分巡回路に含まれる都市数を知る必要がある．5-2 では 5-3 での探索コスト削減のため最も小さな部分巡回路を選択している．5-3 では探索コスト削減のため全探索は行わず，枝 e' を e に地理的に近いものに制限して探索しても解の精度は劣化しない．[永田 99, 永田 06] では Fig. 図 2(b) に示されるように，それぞれの枝 $v_1v_2(=e)$ に対し，$v_3v_4(=e')$ として，v_3 または v_4 の少なくとも一方が v_1 または v_2 の一方から i 番目までに近い都市であるような枝のみを探索している．本論文では先行研究と同様に $i = 10$ とする．

2·2 *E-Set* の構成法

EAX の基本ステップ 3 において，*E-set* は任意の *AB-cycle* の組み合わせで構成することができる．単純な選択方法として，次の 2 つの方法が提案されている．

EAX-Rand: *AB-cycle* を確率 1/2 でランダムに選択して *E-set* を構成する [永田 99].

EAX-1AB: *AB-cycle* を任意に 1 つ選択して *E-set* を構成する [永田 00, 永田 06].

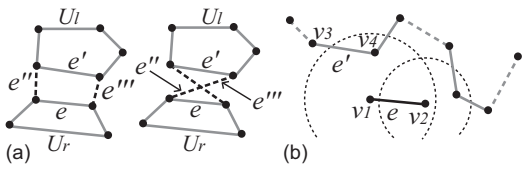

図 2 EAX の基本ステップ（ステップ 5-3）

EAX-Rand を用いた場合，生成される中間個体は tour-A と tour-B の枝を均等に含む（tour-A に対するオーダー N の本数の枝の交換で生成される）傾向にあり，修正操作後に得られる子個体も同様の傾向を持つ．本論文ではこのような交叉を大域的 EAX と呼ぶことにする．

一方，EAX-1AB を用いた場合，生成される中間個体は tour-A に類似しており（tour-A に対する少数の枝の交換で生成される），修正操作後に得られる子個体も同様の傾向を持つ．本論文ではこのような交叉を局所的 EAX と呼ぶことにする．

3. 局所的 EAX の高速実装法

本章では局所的 EAX の高速実装法を提案する．まず，従来実装*3 の概要と高速実装の概要について述べる．次に，高速実装のためのデータ構造とアルゴリズムを提案し，計算量について述べる．

3·1 従来実装の概要

GA 集団内の各個体（巡回路）は *doubly linked list* で表現される．*doubly linked list* はサイズが $N \times 2$ の配列 *Link* で表され，$Link[i][0]$ と $Link[i][1]$ はそれぞれ都市 i に隣接する 2 都市を表す．ただし，それらが [0] あるいは [1] のどちらに入るかは任意で良い*4．$Link_A$，$Link_B$ をそれぞれ E_A，E_B に対する *doubly linked list* とする．

E_C も *doubly linked list* で表現され，これを $Link_C$ で表すものとする．例えば，EAX の基本ステップ 4 で $E_C := E_A$ を実行する際，$Link_A$ の要素がすべて $Link_C$ へコピーされる．

提案する高速実装の従来実装に対する本質的な差異は，ステップ 5-1 を高速化するためのデータ構造とアルゴリズムにある．記述の重複を避けるため，従来実装についてはステップ 5-1 のアルゴリズム（データ構造は $Link_C$）についてのみ記述する．それ以外のステップのアルゴリズムとデータ構造については細かい点を除き，本質的には 3·3 節で述べる高速実装の場合と同様である．

ステップ 5-1 では中間個体を構成する部分順回路の個数 m と各部分巡回路に含まれる都市の数 A_l $(l = 1, \ldots, m)$

*3 [永田 99] で用いられた実装法．その後の EAX の関連研究 [Ikeda 02, Chan 05, 永田 06] においても，GA の計算時間は改善されていない．

*4 本来の定義では任意に定められた巡回路の方向対して，[0] には直前，[1] には直後の都市が入る．

をカウントする．ステップ 4 終了後，$Link_C$ で表現された中間個体が得られるが，ある部分順回路 l に含まれる都市の個数 A_l を知るためには，その部分順回路に含まれる任意の都市からスタートしてその都市に戻るまで $Link_C$ に従い部分順回路をたどる必要がある．例えば，中間個体がたまたま一つの順回路で構成されていた場合でも，全ての都市をたどらなければ中間個体が一つの順回路であることを確認することはできない．よって，ステップ 5-1 では，まだたどられていない都市をスタート地点として，上記の操作を全ての都市がたどられるまで繰り返す．従って，このステップの計算量は $O(N)$ となる．

3･2 高速実装の概要

先行研究 [永田 99, 永田 06] で EAX を用いる場合，一組の両親から 30–100 個の子個体を生成する．EAX の基本ステップ 1 と 2 を合わせた計算量は $O(N)$ であるが，各両親に対して一回だけ実行すれば良いので，従来実装において計算時間に与える影響はほとんどない．一方，ステップ 3–6 は一子個体が生成される毎に実行される．大域的な交叉を用いる場合は，オーダー N の本数の枝の交換が行われるため，これらのステップの計算量を $O(N)$ より小さくすることは原理的に不可能である．一方，従来実装において局所的 EAX を実行する場合，ステップ 5-1 のみが計算量 $O(N)$ を要し，それ以外のステップは局所的な操作（$O(N)$ より少ない計算量）で実行可能である（詳細は 3･4 節で述べる）．それゆえ，従来実装においてステップ 5-1 が局所的 EAX を実行する際のボトルネックとなっている．

中間個体が局所的 EAX によって tour-A から少数の枝を取り除き同数の枝を付け加えられて生成される場合，これらの交換に用いた枝の集合，すなわち *E-set* の情報を用いることでステップ 5-1 の計算量を削減することができる．提案する局所的 EAX の高速実装ではこの考え方がアイデアの中心となる．

従来実装では基本ステップ 4 で $E_C := E_A$ を実行するための計算量が $O(N)$ である．局所的 EAX の高速実装の補助的な工夫として，$Link_A$ を直接 $Link_C$ として用い（$Link_A$ が子個体として直接更新される），子個体が一つ生成された後はそれを Undo して（tour-A に戻して）次の子個体生成に用いる．子個体を記録しておく必要がある場合は，元の $Link_A$ からの変更箇所のみを記録しておけばよい．この工夫は LK 法などの局所探索では基本的なものであるが，局所的 EAX においてステップ 5-1 の高速化がなされていない場合には有意な効果を持たず，実装する必要がなかった．

3･3 局所的 EAX の高速実装法

本節では局所的 EAX の高速実装のためのデータ構造とアルゴリズムを示す．アルゴリズムでダッシュが付されたステップが高速実装に関連した箇所である．

§1 データ構造

従来実装と同様，GA 集団内の各個体（巡回路）は *doubly linked list* で表現され，E_A と E_B に対する *doubly linked list* をそれぞれ $Link_A$，$Link_B$ とする．3･2 節で述べたように，交叉の際には E_C として $Link_A$ を直接用いるが，さらに以下に述べるデータ構造を併用する．図 3 に tour-A，ステップ 4 で適用される *E-set*，その結果生成される中間個体の例を示している．

(a) に tour-A の巡回路を表すデータ構造である *array* 表現を示す．*array* 表現では巡回路はサイズ N の配列 $City$ で表され，$City[p]$ は p 番目に訪れる都市を表す．ただし，スタートとなる都市は任意でよい．さらに，$Pos[City[p]] = p$ $(p = 1, \ldots, N)$ となる配列も用意する．

(b-1) に中間個体を表現するデータ構造の概念図を示す．図において配列 $City$ は幾つかのブロックに分離されているが，*E-set* に含まれる tour-A の枝を tour-A から取り除いた結果に対応し，各ブロックを *Segment* と呼ぶことにする．各 *Segment* をつないでいる線は *E-set* に含まれる tour-B の枝を tour-A に付け加えた結果に対応している．よって，*Segment* の個数は k_1（*E-set* に含まれる tour-A の枝の数）となる．この関係を表現するデータ構造を (b-2) に示す．各 *Segment* ごとに *Segment I.D.* が任意に割り振られる．各 *Segment* は $City$ 上での両端の位置 (*Beginning position*, *End position*)，tour-B の枝による接続先の都市の $City$ 上での位置 (*Adjacent position*)，属している部分巡回路の番号 (*Sub-tour I.D.*) を保持する．このデータ構造は LK 法で用いられている巡回路のデータ構造の一つである *Two-Level Tree* [Fredman 95] と本質的には同じである．

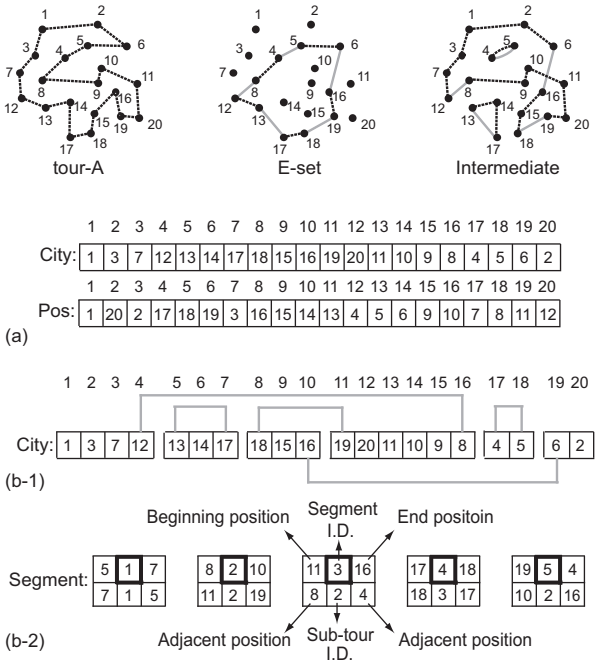

図 3 中間個体を表現するためのデータ構造

§2 アルゴリズム（基本ステップ）

1 G_{AB} として，$Link_A$ と $Link_B$ を直接用いる．

1' tour-A に対し，*City* と *Pos* を生成.

2 $Link_A$ の枝 $Link_B$ の枝を交互に重複なくランダムにたどることで全ての *AB-cycle* を生成する．

3 *AB-cycle* を手法に応じて選択する（*E-set* を構成）．

4 枝の交換に対応して $Link_A$ を更新する．例えば, 枝 $v_1v_2(\in E\text{-}set \cap E_A)$, $v_2v_3(\in E\text{-}set \cap E_B)$ に対して,
$Link_A[v_2][0] := v_3$; (if $Link_A[v_2][0] = v_1$)
$Link_A[v_2][1] := v_3$; (if $Link_A[v_2][1] = v_1$)
などと更新される．

4' *E-set* に含まれる tour-A の枝の両端の都市（k_1 個のペア）を *Pos* に従い小さい順にソートする．ソートされた値から順に 2 つずつ取りだすことで各 *Segment* の両端（*Beginning position*, *End position*) を得る．各 *Segment* に対し，*Segment I.D.* を任意に与え，*Adjacent position* を *E-set* に含まれる tour-B の枝に従い記録する．

5 3·3·3 節に述べる．

6 生成子の数が指定数に達していれば終了．そうでなければ，$Link_A$ を Undo して，ステップ 3 へ．

§3 アルゴリズム（修正操作）

m を中間個体を構成する部分巡回路の個数，A_l $(l = 1, \ldots, m)$ を l 番目の部分巡回路に含まれる都市を表す変数とする．S をサイズ N の配列として，デフォルトでは 0 が入っているものとする．

5-1' 各 *Segment* の *Beginning position*, *End position*, *Adjacent position* に従って部分巡回路をたどることで同一の部分巡回路に属する *Segment* を同定し，*Sub-tour I.D.* に記録する（部分巡回路番号は任意に決めて良い）．この過程で m と A_l $(l = 1, \ldots, m)$ が得られる．

5-2 最小の A_l を持つ部分巡回路の番号を r とする．また，この部分順回路に含まれる都市の集合を V とする（*Segment* 表現から容易に得ることができる）．$S[v] = 1$ $(v \in V)$ とする．

5-3 $\{e^*, e'^*, e''^*, e'''^*\}$ の探索において，$e(= v_1v_2) \in U$, $e'(= v_3v_4) \in E_C \backslash U$ ペアの生成を以下のように行う．
$\forall v_1 \in \{v | v \in V\}$,
$v_2 \in \{Link_A[v_1][0],\ Link_A[v_1][1]\}$,
$v_3 \in \{v | v \in Near(v_1, 10),\ S[v] \neq 1\}$,
$v_4 \in \{Link_A[v_3][0],\ Link_A[v_3][1]\}$,
ただし $Near(v, 10)$ は都市 v から 10 番目までに近い（自身は含まない）都市の集合とする．

5-4 枝の交換に対応して $Link_A$ を更新する（更新法はステップ 4 と同様）．

5-5' *Beginning position* $\leq Pos[v_3^*] \leq$ *End position* となる *Segment* の *Sub-tourI.D.* を r' とする．*Sub-tour I.D.* が r (または m) である *Segment* 全てについて，*Sub-tour I.D.* を r' (または r) とする．$A_{r'} := A_{r'} + A_r$, $A_r := A_m$ とする．S を Undo する（すべて 0 にする）．$m := m - 1$ として，m が 1 なら $Link_A$ が巡回路となり終了．そうでなければ 5-2 へ．

3·4 計　算　量

提案した高速実装法における各基本ステップの (最悪) 計算量について述べる．

まず，以下の定義を行う．以下の値はステップ 3–6 の一サイクルにおける値とする．

k_1: ステップ 4 で交換される枝の数 $(= |E\text{-}set|/2)$.
k_2: ステップ 5-4 で交換される枝の数 $(= 2 \cdot (k_4 - 1))$.
k_3: ステップ 5-2 で選ばれた A_r の総和.
k_4: 中間個体を構成する部分巡回路の個数 $(\leq k_1)$.

ステップ 1 と 2 はそれぞれ計算量が $O(N)$ となる．ステップ 3 の計算量は高々生成された *AB-cycle* の個数のオーダーであり，$O(N)$ にくらべ十分小さい（生成する *AB-cycle* の個数に上限をもうけても良い）．ステップ 4 の計算量は $O(k_1)$ である．ステップ 4' の計算量は $O(k_1^2)$ となるが，これは，k_1 個の都市の *Pos* の値に対するソートのためである．ステップ 5 の計算量は後述する．ステップ 6 の計算量は $O(k_1 + k_2)$ である．

ステップ 5 を一回実行する際，5-2 – 5-5 は $(k_4 - 1)$ 回呼び出されるが，それぞれを総合した計算量について述べる．ステップ 5-1 の計算量は $O(k_1)$ となるが，これは，*Segment* に従い部分巡回路をたどるためである．ステップ 5-2 の計算量は $O(k_4^2 + k_3)$，ステップ 5-3 の計算量は $O(k_3)$，ステップ 5-4 の計算量は $O(k_2)$ である．ステップ 5-5 の計算量は $O(k_1k_4 + k_3)$ となるが，これは r' の検索と U の Undo のためである．

局所的 EAX として EAX-1AB を用いた場合，実験的には $k_1 \ll N$ となることが多い．形式的には，局所的 EAX の定義を k_1 が N に対して十分小さな *E-set* を用いるものとすれば良い．また，$k_2 = 2 \cdot (k_4 - 1)$，$k_4 \leq k_1$ であるので $k_2, k_4 \ll N$ となる．k_3 も実験的には $k_3 \ll N$ となることが多い．この理由は，EAX-1AB のような局所的 EAX を用いた場合，中間個体は図 1 の左側に示されているような N 本の枝の大半を占める 1 つの大きな部分巡回路とそれ以外の（一般には複数の）小さな部分巡回路から構成される傾向があるからである．ここで，k_3 はステップ 5-3 で探索される $v_1 \in V$ の延べ個数であり，最も大きな部分巡回路は V として考慮する必要がないことに注意する．形式的には，ステップ 5-3 において $v_1 \in V$ を全探索せずに，各 V ごとに v_1 の探索を定数個に限定し，さらに，v_3 が V に含まれていないことを判定する際に Ur を用いない (V を構成している *Segment* をそのつど確認（最悪 k_1 回の計算）) ものとすれば，各ステップの計算量から k_3 を排除できる．この時，ステップ 5-2 の計算量は $O(k_4^2)$，ステップ 5-3 の計算量は $O(k_1k_4)$，ステップ 5-5 の計算量は $O(k_1k_4)$ となる．

以上の考察により局所的 EAX を用いた場合，形式的

にはステップ 3–6 を一サイクル実行するのに要する計算量は $O(k_1^2)$ となる．ステップ 1 と 2 の計算量がそれぞれ $O(N)$ であるが，これは各両親に対し一回だけ実行すれば良い．ただし，これは従来実装の場合とは異り，高速実装では無視できない計算コスト（EAX 全体の 20–30% 程度）となる．実際の計算機実験ではステップ 5-3 における計算コストが最も大きい（EAX 全体の 60–80% 程度）．

4. 局所的な計算による多様性維持

本章では 3 章で提案した局所的 EAX の高速実装を用いた場合でも，相対的にわずかな計算コストの増加で効果的に多様性維持を行う方法を提案する．

4·1 TSP に対する多様性指標

前川らの研究では TDGA と呼ばれる多様性維持手法を提案し TSP へ適用した [前川 97]．TDGA では，TSP を探索する GA 集団の多様性指標として，次式で定義される枝エントロピー H を提案している．本論文でも，枝エントロピーを集団の多様性指標として用いる．

$$H_i = -\sum_{j=1}^{N} P_{ij}\log(P_{ij})\ (i=1,\ldots,N) \qquad (1)$$

$$H = \sum_{i=1}^{N} H_i = -\sum_{i=1}^{N}\sum_{j=1}^{N} P_{ij}\log(P_{ij}) \qquad (2)$$

，ただし H_i は都市 i に関するエントロピーを表し，P_{ij} は集団中の個体において都市 i が j と接続する割合を表す．ただし，各都市は巡回路において 2 つの接続先をもつため P_{ij} の計算では 1/2 を乗じている．集団のエントロピーは (2) 式で定義される．これは，各都市 i において P_{ij} が独立，かつ集団サイズが無限大と仮定した場合における集団の近似的なエントロピー指標と言える．

4·2 多様性維持法

§1 世代交代モデル

procedure GA(N_{pop}, N_{ch})
begin
1: 初期集団 $\{x_1, x_2, \ldots, x_{N_{pop}}\}$ の生成
2: **repeat**
3: $i \in \{1,\ldots,N_{pop}\}$ をランダムに集団に割り振る;
4: **for** $i := 1$ **to** N_{pop} **do**
5: $p_A := x_i$; $p_B := x_{i+1}$; $(x_{N_{pop}+1} = x_1)$
6: $\{c_1, c_2, \ldots, c_{N_{ch}}\} := \mathrm{EAX}(p_A,\ p_B)$;
7: $x_i := \mathrm{Select_Best}(c_1, c_2, \ldots, c_{N_{ch}})$;
8: **end for**
9: **until** 終了条件が満たされる;
10: **return** 集団中の最良個体;
end

procedure GA において，N_{pop} と N_{ch} は指定されるパラメーターで，それぞれ集団サイズ，各両親から生成される子個体の数である．Line 1 では初期集団が生成される．Line 6 では EAX を用いて各両親 p_A, p_B から N_{ch} 個の子個体が生成される．p_A と p_B は EAX の手続きにおいてそれぞれ tour-A と tour-B に対応するものとする．Line 7 では N_{ch} 個の生成子から" 最良" の個体を選択し，集団中の個体 x_i と置換える．ただし，子個体の評価は 4·2·2 節で述べられる評価関数により行われるものとする．また，x_i よりも巡回路長の短い子個体が生成されなかった場合は x_i がそのまま残るものとする．局所的 EAX を用いた場合 $x_i(=p_A)$ に類似した子個体が生成されるため，x_i のみを生成子個体で置換えることは，多様性維持の面で優れているといえる．

§2 評価関数

L を集団の巡回路長の平均値，H を集団の枝エントロピーと定義する．y を両親 $p_A(=x_i)$，$p_B(=x_{i+1})$ から生成された子個体とし，$\Delta L(y)$ と $\Delta H(y)$ をそれぞれ x_i を y で置換えた際の集団の L と H の変化と定義する．集団全体の巡回路長を短くする観点からは $\Delta L(y)$ の値は小さい方が好ましいので，y の評価を以下のように定義する方法を Greedy 選択と呼ぶことにする．ただし，評価値は大きい方が良いものとする．

$$Eval_{Greedy}(y) = -\Delta L(y) \qquad (3)$$

一方，多様性維持の観点からは $\Delta H(y)$ の値は大きい方が望ましい．そこで，$\Delta L(y)$ と $\Delta H(y)$ のトレードオフを考慮して y を評価する必要がある．提案手法では以下の式で子個体を評価し，最も大きな値を持つ子個体を x_i と置換えるものとする．

$$Eval_{Ent}(y) = \begin{cases} \frac{\Delta L(y)}{\Delta H(y)} & (\Delta L < 0, \Delta H < 0) \\ -\frac{\Delta L(y)}{\epsilon} & (\Delta L < 0, \Delta H \geq 0) \\ 0 & (\Delta L \geq 0) \end{cases} \qquad (4)$$

，ただし ϵ を十分小さな正定数とする．

x_i が子個体 y で置換えられる場合は必ず $\Delta L(y) < 0$（巡回路長が短くなる）であるが，この時 $\Delta H(y) < 0$（集団の多様性が失われる）となる傾向がある．この場合は単位多様性損失当りの巡回路長の改善量である $\frac{-\Delta L(y)}{-\Delta H(y)}$ により子個体 y を評価することとした．だだし，$\Delta L(y) < 0$ かつ $\Delta H(y) \geq 0$ となる子個体が存在する場合には，多様性の損失はないものとみなし，そのような個体の中で $\Delta L(y)$ の値が最も小さい子個体が最も良い評価を得る．

§3 先行研究との相違

提案した多様性維持法は，先行研究である [永田 06] において多様性の定義を枝エントロピーへと変更したものである．先行研究では以下の評価関数を用いた．

$$Eval_{LDL}(y) = \begin{cases} \frac{\Delta L(y)}{\Delta D(y)} & (\Delta L < 0, \Delta D < 0) \\ -\frac{\Delta L(y)}{\epsilon} & (\Delta L < 0, \Delta D \geq 0) \\ 0 & (\Delta L \geq 0) \end{cases} \qquad (5)$$

，ここで $\Delta D(y) = d(x_i, y) - d(x_i, x_{i+1})$，$d(a, b)$ は 2 個体 a, b の間で異る枝の数と定義される．この評価関数では，子個体 y で x_i を置換えた際の集団の多様性の変化を $\Delta D(y)$，すなわち集団中の 2 個体 x_i と x_{i+1} がどれだけ近づくかという局所的な変化により定義した．これを局所的多様性の損失（Local diversity loss, LDL）と呼ぶ．

4·3 $\Delta H(y)$ の計算法

局所的 EAX の高速実装を用いた場合でも，**procedure** GA を用いた場合には，$Eval_{Ent}(y)$ の計算はわずかな計算コストで行うことがきる．以下に $\Delta H(y)$ の計算法を示す．また，$\Delta L(y)$ の計算についても示しておく．

(2) 式で示した枝エントロピーは (6) 式ように書ける．X は集団中に存在する枝の集合，$F(e)$ は集団中で枝 e $(\in X)$ を持つ個体の数，N_{pop} は集団サイズである．

$$H = -\sum_{e \in X} F(e)/N_{pop} \log(F(e)/N_{pop}) \qquad (6)$$

E_{ad} を p_A から y を生成する際に付け加えられた枝の集合，E_{re} を取り除かれた枝の集合とする．これらの集合は EAX の手続き中に記録しておく．$\Delta L(y)$ と $\Delta H(y)$ は以下のように計算される．ただし，$w(e)$ を枝 e の長さ，$A(x) = -x/N_{pop} \log(x/N_{pop})$ $(A(0) = 0)$ とする．

$$\Delta L(y) = \frac{1}{N_{pop}} \{ \sum_{e \in E_{ad}} w(e) - \sum_{e \in E_{re}} w(e) \} \quad (7)$$

$$\Delta H(y) = \sum_{e \in E_{ad}} \{ A(F(e)+1) - A(F(e)) \} + \sum_{e \in E_{re}} \{ A(F(e)-1) - A(F(e)) \} \quad (8)$$

式 (8) の計算には $F(e)$ が必要となるが，これは GA の初期集団を生成した時点で一度計算しておく．また，**procedure** GA (line 7) において，x_i が y で置換えられた際には，$F(e) := F(e) + 1$ $(e \in E_{ad})$，$F(e) := F(e) - 1$ $(e \in E_{re})$ として $F(e)$ を更新すれば良い．

E_{ad} と E_{re} の要素数は交叉で交換される枝の数であるので，局所的 EAX を用いた場合，$Eval_{Ent}(y)$ の評価に要する計算量は $O(k_1 + k_2)$ となり局所的に計算できる（k_1 と k_2 の意味は 3·4 節を参照）．実際の計算機実験では，いずれのベンチマークを用いた場合でも $Eval_{Ent}(y)$ の計算に要する計算コストは GA 全体の 5%以下であった．一方 TDGA の枠組みでは，生成した子個体が集団中の任意の個体と置換えられるため，ここで述べたような計算の効率化はできず，多様性の計算が GA 全体の計算時間のほとんどを占めてしまう．

5. 計 算 機 実 験

本章では，3 章で提案した局所的 EAX の高速実装の効果の検証，4 章で提案した枝エントロピーを用いた多様性維持の効果の検証を行う．

5·1 実 験 設 定

以下の GA は局所的 EAX を用いた GA の構成で，これを GA-Local と呼ぶ．

[GA-Local]

- 局所的 EAX として EAX-1AB を用いる（line 6）．
- **procedure** GA（4·2·1 節）を用い，$N_{ch} = 30$ とする．N_{pop} は適宜設定．EAX-1AB を用いた場合，$N_{ch} = 30$ 程度が良いと報告されている [永田 06]．
- 初期集団は，2-opt 法 [Johnson 97] による局所探索を用いて生成する（**procedure** GA，line 1）．
- 子個体の選択（line 7）に $Eval_{Greedy}$，$Eval_{LDL}$，$Eval_{Ent}$ のいずれかを用いる．それぞれ GA-Local (Greedy)，GA-Local (LDL)，GA-Local (Ent) と表記する．
- 集団中の最短巡回路長の改善が 50 世代にわたり停滞した時点での世代数を G として，$G/10$ 世代にわたり最短巡回路長の改善が停滞した場合（$50 > G/10$ ならその時点），交叉を EAX-5AB に切換える（line 6）．EAX-5AB とはランダムに 5 個の *AB-cycle* を選択して *E-set* を構成する EAX である．
- EAX-5AB で 50 世代にわたり最良解の改善が停滞した場合は終了 (line 9).

GA-Local の設定では探索の大半は EAX-1AB を用い，終盤のみ EAX-5AB を用いる．EAX-1AB を用いた GA で探索が停滞した場合，集団内の各個体は深い局所解に陥っているため，EAX-1AB のような局所的な交叉ではもはや解を改善することは困難になる．そこで，探索終盤はより大域的な交叉（EAX-5AB）を用いることで，さらに集団内の解候補を改善することができる．

比較のため，大域的 EAX を用いた GA による実験も行う．以下の GA の構成を GA-Global と呼ぶ．ただし，GA-Local と異なる設定の部分のみを記述する．

[GA-Global]

- 大域的 EAX として EAX-Rand を用いる（line 6）．
- **procedure** GA を用い，$N_{ch} = 50$ とする．N_{pop} は適宜設定．EAX-Rand を用いた場合，$N_{ch} = 50$ 程度が良いと報告されている [永田 06]．
- 子個体の選択（line 7）に $Eval_{Greedy}$ を用いる．この設定を GA-Global (Greedy) と表記する．
- 集団の最良順回路長と平均順回路長の差が 0.1 以下になったら終了 (line 9)（通常，この時点での最良解の停滞世代数は 50 以下である）．

適用問題は [TSPLIB] の中から選んだ 1084 – 18512 都市規模の 14 問題とし，これらは表 1 に示されている．各問題で 20 試行を行う．なお，実装は C++を用い，3.06 GHz Xeon Processor 2 GB RAM 上で実行した．

5·2 局所的 EAX の高速実装の効果

本節では 3 章で提案した局所的 EAX の高速実装の効果を検証する．検証法として GA-Local (Greedy) の設定

表 1 GA-Local または GA-Global で $Eval_{Greedy}$，$Eval_{LDL}$ あるいは $Eval_{Ent}$ を子個体の選択に用いた場合の探索性能の比較．'opt.' は最適解を得た試行回数，'err.' は各試行で得た最良解の最適解からの誤差の平均，'gen.' は各試行で最良解を発見するまでに要した世代数の平均，'time(s)' は一試行当りに要する CPU 時間の平均（秒）（3.06 GHz Xeon single processor で計測）である．各問題に対し 20 試行を行った．

		GA-Global(Greedy) [Npop = 300]				GA-Local(Greedy) [Npop = 300]				GA-Local(Local) [Npop = 300]				GA-Local(Ent) [Npop = 200]			
instance	no. of cityes	opt.	err.(%)	gen.	time(s)	opt.	err.(%)	gen.	time(s)	opt.	err.(%)	gen.	time(s)	opt.	err.(%)	gen.	time(s)
mv1084	1084	15	0.0042	21	35	16	0.0014	44	30	16	0.0020	57	45	20	0.0000	79	36
pcb1173	1173	7	0.0122	24	51	10	0.0029	60	26	18	0.0005	78	37	20	0.0000	127	37
u1432	1432	7	0.0199	19	156	2	0.0356	61	54	9	0.0116	89	63	14	0.0035	102	48
vm1748	1748	12	0.0145	26	97	8	0.0268	80	55	19	0.0017	86	71	20	0.0000	134	69
pr2392	2392	6	0.0116	28	216	11	0.0045	101	63	18	0.0007	150	94	20	0.0000	233	100
pcb3038	3038	0	0.0166	39	520	1	0.0091	158	162	6	0.0034	242	260	20	0.0000	364	247
fnl4461	4461	0	0.0565	67	2432	1	0.0112	354	540	9	0.0015	483	902	18	0.0001	795	780
rl5915	5915	0	0.0135	32	838	1	0.0075	115	293	7	0.0023	168	540	15	0.0007	289	469
pla7397	7397	0	0.0307	76	4879	0	0.0065	992	761	0	0.0051	1083	942	0	0.0066	1148	878
rl11849	11849	0	0.0211	71	5879	0	0.0075	482	1315	2	0.0015	734	1957	12	0.0004	1236	2728
usa13509	13509	0	0.0720	148	15003	0	0.0068	1070	3486	0	0.0016	1771	5252	0	0.0011	3081	6344
brd14051	14051	0	0.0937	172	24018	0	0.0098	1402	4890	0	0.0031	2254	8000	2	0.0018	3179	9421
d15112	15112	0	0.1014	186	28359	0	0.0065	1773	8162	0	0.0017	2717	13409	0	0.0017	4531	11440
d18512	18512	0	0.1034	226	34863	0	0.0077	2181	9617	0	0.0035	3526	13989	0	0.0024	5391	16404

において，従来実装と高速実装の計算時間の比較を行った．従来実装は 3･3 節で提案した高速実装から 3･2 節で述べた高速実装の工夫を取り除いたものをさす．また，参考として大域的 EAX に対する高速実装の効果も検証する．検証法として，GA-Global (Greedy) の設定において，従来実装と高速実装の計算時間の比較を行った．なお，全ての GA で集団サイズを $N_{pop} = 300$ とした．

図 4 に実験結果を示す．グラフは都市サイズに対する一試行当りの平均 CPU 時間を両対数グラフで示している．また，各グラフを x を都市数 y を CPU 時間として，$y = 10^b x^a$ によりフィッティングした結果についても示す．これは $X = \log x$，$Y = \log y$ の変換に対し，$Y = aX + b$ に対する最小自乗法により行った．

GA-Local では高速実装により a の値は 2.8 から 2.0 まで減少しており，例えば 10,000 都市以上の問題では計算コストは約 1/10 – 1/15 に削減した．一方，GA-Global も高速実装により計算コストが減少するが，効果は小さく，a の値は 2.4 から 2.3 まで減少した．例えば 10,000 都市以上の問題では計算コストは約 1/2 – 1/3 に削減した．

大規模問題において，従来実装の GA-Global は GA-Local よりも元々の計算コストが少ないが，これは GA-Global の方が多様性を急速に失いやすいためである*5．そのため，最終的に得られる解の質は GA-Local の方が良い．参考として表 1 には GA-Local (Greedy) と A-Global (Greedy) で得られた解の質についてに示す．ただし，高速実装で得られた結果のみ示している（従来実装でも変わらない）．

*5 この理由については [永田 06] を参照されたい．実際，計算時間を同一（従来実装）とした場合の比較において，GA-Local の方が GA-Globl よりも質の良い近似解を得る．

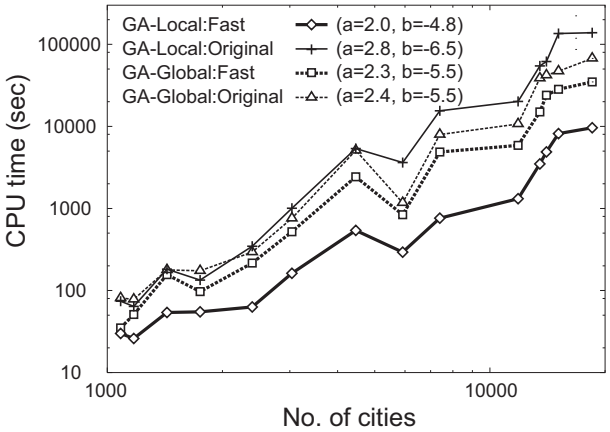

図 4 GA-Local (Greedy) と GA-Global (Greedy) における従来実装と高速実装の比較．横軸は都市サイズ，縦軸は一試行に要する平均 CPU 時間で，両対数グラフで示している．また，a，b は $y = 10^b x^a$ によって各グラフをフィッティングした値である．

5･3 選択法の効果

本節では 4 章で提案した多様性維持法の効果を検証する．検証法として，GA-Local (Greedy)，GA-Local (LDL)，GA-Local (Ent) の比較を行う．

集団サイズは GA-Local (Greedy) と GA-Local (Local) では $N_{pop} = 300$ とするが，GA-Local (Ent) では $N_{pop} = 200$ とした．これは，GA-Local (Ent) では収束に要する世代数が増加するため，GA-Local (LDL) とほぼ同じ計算時間で比較できるよう集団サイズを少なくしたためである．なお，GA-Local (Greedy) は集団サイズを増やして GA-Local (LDL) と同一計算時間をあたえても，これと同等の性能は得られないことが知られている [永田 06]．

表 1 に実験結果を示す．GA-Local (Ent) を用いた場合，ほぼ全ての問題において最適解発見回数と近似解の誤差の平均が最も優れていることが分かる．特に GA-Local

(Ent) は brd14051 のような大規模問題でも最適解を発見することに成功している．参考として，GA-Local (Ent) に要する計算時間を 5･2 節と同様に x を都市数 y を CPU 時間として，$y = 10^b x^a$ によりフィッティングを行った結果，$a = 2.2$，$b = -5.1$ となった．

6. 他手法との比較

本章では，最終的な提案手法である GA-Local (Ent) と TSP の state-of-the-art 近似解法との比較を行い，提案手法の近似性能を検証する．

比較する TSP の近似解法として，1 章で述べた ILK [Johnson 97]，CLK [Cook 00]，HLK [Helsgaun 00]（以上 3 つは LK 法に基づいた Iterated local search），HeSEA [Tsai 04]（EAX を用いた GA と LK 法のハイブリッド）を対象とする．GA-Local (Ent) は集団サイズを $N_{pop} = 200$ とすると，局所探索に比べ多くの計算時間を要するため，$N_{pop} = 30$ とした場合の実験結果についても示す．

表 2 に比較結果を示す．HeSEA のデータは文献 [Tsai 04] から抜粋した．ILK，CLK，LKH のデータは “8-th DIMACS Implementation Challenge: The Traveling Salesman Problem” [DIMACS] *6 の結果から抜粋した．ILK，CLK，LKH のデータは，なるべく集団サイズ 30 の GA-Local (Ent) と（マシン性能の差を考慮して）計算時間が近くなる（Iterated local search における）イテレーション回数のデータを選択した．ただし，LKH についてはより多くのイテレーション回数での実験データも抜粋した．各手法の試行回数，計算機性能については表 2 に示されている．

集団サイズ 30 の GA と ILK の比較では，計算コストの大小関係がまちまちであるが，約半数の問題で GA が ILK よりも計算コストと解の質の両面で優れている．

集団サイズ 30 の GA と CLK との比較では，解の質では GA が優位であるが，計算コストは（特に大規模問題で）CLK が少ない傾向にある．

集団サイズ 30 の GA と LKH[Helsgaun-N/10] との比較では，問題規模が小さい場合，GA は計算コストと解の質で LKH に劣る．大規模問題では計算コストは同程度か GA がやや少ないものの，得られる解の質では LKH の方がやや良い．

集団サイズ 200 の GA と HeSEA を比較すると，両者の計算コストはほぼ同程度であるが，特に問題規模が大きい場合に GA の方が近似精度に優れていると言える．

集団サイズ 200 の GA と LKH[Helsgaun-N] との比較を行う．この設定は，“8-th DIMACS Implementation Challenge: The Traveling Salesman Problem” において，平均的に最も質の高い解を得た手法である．問題規模が小さい場合は GA の優位性は見られないものの，大規模問題において，GA が計算コストと解の質の両面で優位である．

また，[Cook 00] では CLK のイテレーションを 24 時間（CPU: 300 MHz Pentium II，Xeon に比べ約 10 倍遅い）にした実験を行っているが，最終的に得られる解の誤差は 0.190% (rl11849)，0.092% (usa13509)，0.066% (brd14051)，0.064% (d15112)，0.062% (d18512) であった（これより小さな問題ではデータなし）．問題 rl11849，usa13509，rd14051 では計算コストと解の質の両方で集団サイズ 200 の GA が優れている．d15112，d18512 では GA の計算時間の方が大きいが，両者の近似解の誤差を考慮すると CLK の計算時間を GA と同じにしても GA の方が精度の良い近似解を得られるものと予想される．

以上の結果から，一試行に与えられる計算時間が集団サイズ 30 の GA を実行できる程度である場合は，GA の近似精度は ILK，CLK に匹敵し，LKH に対しては劣っているといえる．一方，与えられる計算時間が比較的長く，集団サイズ 200 の GA を実行できる程度である場合には，都市数が 10,000 以上の大規模問題において GA は LK 法に基づく TSP の state-of-the-art 近似解法よりも優れていると結論できる．

7. 考　　察

6 章では，提案手法である GA-Local (Ent) が LK 法に基づく TSP の state-of-the-art 近似解法と比較しても良い探索性能を持つことを示した（表 2）．本章ではその要因を関連研究を含めて考察する．

TSP の場合に限らず，GA と局所探索を比較した場合の GA の問題点の一つは計算時間を比較的多く要することある．これには主に二つの理由が考えられる．

(1)交叉は局所探索オペレーターと比較して良い解を生成しにくい．

(2)局所探索オペレーターと比較して交叉自体に計算時間がかかる．

(1) に関する研究として，[Nagata 04] では TSP の探索において枝の交換を局所的に制限した交叉は，大域的な枝の交換を行う交叉よりも評価値の良い子個体を生成しやすいことを示した．そのような交叉として，本論文では EAX-1AB を用いた．

本論文では (2) に関する問題点を扱った．交叉を用いて TSP の問題制約を満たした解候補を新に生成する際，一方の親に対する解の変化（交換される枝の数）は小さい方が計算の手間を少なくすることができる．例えば EAX-Rand のような大域的 EAX を用いた場合，一つの子を生成するための計算量は $O(N)$ を下回ることはできない．本論文では EAX-1AB のような局所的 EAX を用いた場合は，一子個体を生成するための計算量を形式的に

*6 TSP の近似解法の コンテスト．主要な近似アルゴリズムのほとんどが参加しており，2002 年までに投稿された結果が掲載されている．これ以降本質的な改善は報告されていない．

表 2 GA-Local (Ent) と HeSEA，ILK，CLK，LKH の探索性能の比較．'err.' は各試行で得た最良解の最適解からの誤差の平均，'time(s)' は一試行当りに要する CPU 時間の平均（秒）である．ただし，GA-Local は 3.06 GHz Xeon 上で計測，HeSEA は 1.2 GHz Pentium IV 上で計測（Xeon に比べ 3 倍程度遅い），ILK，CLK，LKH は 500MHz Alpha processor 換算（Xeon にくらべ 4–5 倍遅い）に規格化された計測時間である．試行回数は，GA と HeSEA が 20 回，ILK，CLK，LKH は 1 回である．

	GA-Local(Ent) [N_{pop}=200]		GA-Local(Ent) [N_{pop}=30]		HeSEA		ILK [JM-N-b10]		CLK [ABCC-10N]		LKH [Helsgaun-N/10]		LKH [Helsgaun-N]	
	err.(%)	time(s)	err.(%)	time(s)	err.(%)	time(s)	err.(%)	time(s)	err.(%)	time(s)	err.(%)	time(s)	err.(%)	time(s)
mv1084	0.0000	36	0.02	5	0.0000	81	0.02	157	0.00	27	0.07	10	0.00	38
pcb1173	0.0000	37	0.04	6	0.0000	85	0.01	66	0.16	20	0.18	9	0.18	24
u1432	0.0035	48	0.09	7	0.0000	107	0.10	94	0.23	31	0.00	12	0.00	67
vm1748	0.0000	69	0.08	10	0.0000	141	0.00	352	0.05	52	0.02	21	0.02	51
pr2392	0.0000	100	0.03	16	0.0000	208	0.11	138	0.42	48	0.00	48	0.00	488
pcb3038	0.0000	247	0.04	36	0.0000	612	0.12	257	0.26	67	0.03	84	0.00	350
fnl4461	0.0001	780	0.04	136	0.0005	2349	0.14	398	0.15	129	0.01	182	0.00	710
rl5915	0.0007	469	0.09	54	0.0001	2773	0.02	1161	0.54	1935	0.04	333	0.04	1118
pla7397	0.0066	878	0.15	102	N/A		0.05	3515	0.27	331	0.02	1568	0.00	9272
rl11849	0.0004	2728	0.06	373	N/A		0.19	2716	0.21	599	0.09	2315	0.06	12291
usa13509	0.0011	6344	0.06	950	0.0074	34984	0.16	4436	0.20	967	0.01	2631	0.01	15470
brd14051	0.0018	9421	0.03	965	N/A		0.18	4195	0.10	646	0.02	8235	0.01	25945
d15112	0.0017	11440	0.03	1625	0.0137	62371	0.20	3408	0.13	981	0.02	7897	0.02	26322
d18512	0.0024	16404	0.04	2106	N/A		0.21	3925	0.15	976	0.03	11163	0.02	60263

は $O(k_1^2)$（k_1 は tour-A から中間個体を生成する際に交換される枝の数）とすることができることを示した（ただし，基本ステップ１，２は除く）．また計算機実験によりその効果を実証した（図 4）．

本論文で提案した GA-Local (Ent) は，GA 集団の多様性維持法として次の二つの利点を持つ．

(i)**procucure** GA において局所的な交叉を用いて子を生成し，類似している方の親が子と置換わることで，集団の変化が小さく多様性維持に優れる．

(ii)**procucure** GA において局所的な交叉を用いることで，集団の多様性の変化を評価するための計算を局所的に行うことができ，無視できる計算コストで効率的に多様性維持を実現できる．

(i) のような考え方は [Ono 98] で提案された CCM モデルに端を発している.(ii) に関し，本論文では多様性指標に枝エントロピーを用いた場合，多様性の計算に関する計算コストは GA 全体の計算コストの 5%以下であることを述べ，GA の近似精度も向上することを示した（表 1）．

ただし，ここまでに述べた提案 GA の利点は全ての組合せ最適化問題に当てはまるものではない．例えば，解候補の評価値を計算すること自体が交叉の手間に較べて大きい場合，GA の問題点の理由 (2) は重要ではない．また，GA 集団の多様性指標として (2) 式のような解を構成する要素の独立した関数を用いることが不適切な場合は，多様性維持法の利点 (ii) は有用とはならない．提案手法のアイデアは，TSP のように解候補を構成する要素（TSP では枝）が評価値に対して比較的独立に作用する場合に特に有効であると考 えられる．

本論文の主張は交叉 EAX において枝の交換を局所化することで GA の探索性能が改善できることを示すことであり，大域的な枝の交換を行う交叉に関する工夫は行わなかった（EAX-5AB という単純な方法を用いた）．一方，[Nagata 06] では，GA の問題点 (1) に関し，大域的な枝の交換を行う EAX を用いた場合に効率的に良い解を生成する方法を考案している．そして，本論文 5･1 節で用いた GA の枠組み (GA-Local (Ent)) において，EAX-5AB の代わりにその交叉を用いることで，GA-Local (Ent) の結果（表 1）を改善することに成功している．例えば，usa13509，brd14051，d15112，d18512 の問題において，同程度の計算時間（一試行当り）で 10 試行中それぞれ，6 回，10 回，4 回，8 回の最適解を発見している．大域的な枝の交換を行う EAX の工夫については今後の論文で扱う予定である．

8. ま と め

本論文では，交叉 EAX で交換される枝が局所化された場合に，交叉の手続きを局所的な計算で効率的に実行する実装法を提案した．また，局所的交叉を用いた場合，枝エントロピーを多様性指標として GA 集団の多様性維持を無視できる程度の計算コストで行うことができることを示した．計算機実験の結果，提案した GA が TSP の state-of-the-art 近似解法に匹敵する性能持つことを示し，特に，都市数が 10,000 を越えるような大規模問題において計算時間が比較的多く与えられた場合に GA が優位であること示した．

最近 30 年間の TSP の state-of-the-art 近似解法がすべて LK 法に基づいた手法であることを考慮すると，提案した GA は TSP の近似解法の進展に大きく貢献するものと言える．

今後は，7 章で考察した提案 GA の利点を，TSP 以外の組合せ最適化問題においても応用し，優れた GA を考案したいと考えている．

◇参 考 文 献◇

[Bonachea 00] D. Bonachea, E.Ingerman, J. Levy and S. McPeak: An Improved Adaptive Multi-Start Approach to Finding Near-Optimal Solutions to the Euclidean TSP, Proc. of the Genetic and Evolutionary Computation Conference, pp. 143–150 (2000).

[Chan 05] C.H. Chan, S.A. Lee, C.Y. Kao and H.K. Tsai: Improving EAX with restricted 2-opt, Proc. of the 2005 Conference on Genetic and Evolutionary Computaion, pp. 1471–1476 (2005).

[Cook 00] D. Applegate, W. Cook and A. Rohe: Chained Lin-Kernighan for large traveling salesman problems, INFORMS Journal on Computing, Vol. 15, No. 1, pp. 82-92 (2003).

[DIMACS] 8-th DIMACS Implementation Challenge: The Traveling Salesman Problem, http://www.research.att.com/ dsj/chtsp.

[Fredman 95] M.L. Fredman, D.S. Johnson, L.A. McGeoch and G. Ostheimer: Data Strctures for Traveling Salesmen, Journal of Algorithms, Vol. 18, No. 3, pp. 432–479 (1995).

[Glover 94] F. Glover: Genetic algorithms and scatter search: Unsuspected potentials, Statistics and Computing, Vol. 4, pp. 131–140 (1994).

[Helsgaun 00] K. Helsgaun: An effective implementation of the Lin-Kernighan traveling salesman heuristic, European Journal of Operational Research, Vol. 126, No. 1, pp. 106–130 (2000).

[Ikeda 02] K. Ideda and S. Kobayashi: Deterministic Multistep Crossover Fusion: A Handy Crossover Composition for GAs, Proc. of the Seventh Int. Conference on Parallel Problem Solving from Nature, pp. 162–171 (2002).

[Johnson 97] D.S. Johnson and L.A. McGeoch: chapter The traveling salesman problem: a case study, Local Search in Combinatorial Optimization, pp. 215–310, John Wiley and Sons (1997).

[Johnson 02] D.S Johnson and L.A. McGeoch: chapter Experimental Analysis of Heuristics for the STSP, THE TRAVELING SALESMAN PROBLEM AND ITS VARIATIONS, pp. 369–438, Kluwer Academic Publishers (2002).

[久保 94] 久保: 巡回セールスマン問題への招待 I, オペレーションズ・リサーチ , Vol. 39, No. 1, pp. 25–31 (2006).

[Lin 73] S. Lin and B. Kernighan: Effective heuristic algorithms for the traveling salesman problem, Operations Research, Vol. 21, No. 2, pp. 498–516 (1973).

[前川 95] 前川, 玉置, 喜多, 西川: 遺伝的アルゴリズムによる巡回セールスマン問題の一解法, 計測自動制御学会誌論文集, Vol. 31, No. 5, pp. 598–605 (1995).

[前川 97] 前川, 森, 玉置, 喜多, 西川: 熱力学的選択ルールを用いた巡回セールスマン問題の遺伝的解法, 計測自動制御学会誌論文集, Vol. 33, No. 9, pp.939–946 (1997).

[Martin 91] O.C. Martin, S.W. Otto, and E.W. Felten: Large-Step Markov chains for the Traveling Salesman Problem, Complex Systems, Vol. 5 No. 3, pp. 299–326 (1991).

[Merz 97] P. Merz and B. Freisleben: Genetic Local Search for the TSP: New Results, Proc. of the 1997 IEEE Int. Conf. on Evolutionary Computation, pp. 159–163 (1997).

[永田 99] 永田, 小林: 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).

[永田 00] 永田, 小林: 遺伝的アルゴリズムを用いた巡回セールスマン問題の解法, 東京工業大学知能システム科学専攻博士論文, (2000).

[Nagata 04] Y. Nagata, Criteria for Designing Crossovers for TSP, Proc. of the 2004 Congress on Evolutionary Computation, pp. 1465–1472 (2004).

[永田 06] 永田: 局所的多様性の損失を考慮した評価関数を用いた GA の TSP への適用, 人工知能学会論文誌, Vol. 21, No. 2, pp. 195–204 (2006).

[Nagata 06] Y. Nagata, New EAX Crossover for large TSP instances, Proc. of the 9th Int. Conf. on Parallel Problem Solving from Nature, pp. 327–381 (2006).

[National TSP] National Traveling Salesman Problems. http://www.tsp.gatech.edu/world/countries.html.

[Ono 98] I. Ono, Y. Nagata, and S. Kobayashi: A Genetic Algorithm taking account of Characteristics Preservation for Job Shop Scheduling Problems, Proc. of the 5th Int. Conf. on Intelligent Autonomous Systems, pp.711–718 (1998).

[Tsai 04] H.K. Tsai, J.M. Yang, Y.F. Tsai, and C.Y. Kao: An Evolutionary Algorithm for Large Traveling Salesman Problem, IEEE Transaction on SMC-part B, Vol. 34, No. 4, pp. 1718–1729 (2004).

[TSPLIB] TSPLIB95, http://www.iwr.uni-heidelberg.de/iwr/compt/soft/TSPLIB95

[Watson 00] J. Watson, C. Poss, D. Whitley et al.:The Traveling Salesrep Problem, Edge Assembly Crossover, and 2-opt, Proc. of the Fifth Int. Conference on Parallel Problem Solving from Nature, pp. 823–833 (2000).

[Whitley 89] D. Whitley, T. Starkweather and D. Fuquay, Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator, Proc. of the 3rd International Conference on Genetic Algorithms, pp. 133–140 (1989).

[山村 92] 山村, 小野, 小林: 形質遺伝性を重視した遺伝アルゴリズムに基づく巡回セールスマン問題の解法，人工知能学会誌, Vol. 10, No. 6, pp.1049–1059 (1992).

〔担当委員：宮崎 和光〕

2006 年 12 月 12 日　受理

著 者 紹 介

永田　裕一（正会員）

1994 年 9 月 東京工業大学応用物理学科卒業, 2000 年 3 月 同大学院総合理工学研究科知能システム科学専攻博士後期課程修了．工学博士．同年 4 月 同大学院総合理工学研究科リサーチアソシエイト，2001 年 4 月より 北陸先端科学技術大学院大学情報科学専攻助手，現在にいたる．遺伝的アルゴリズム，機械学習の研究に従事．