効果的な局所探索制限による Memetic Algorithm の高速化

Efficient Local Search Limitation Strategies in Memetic Algorithm

永田 裕一
Nagata Yuichi

東京工業大学大学院 総合理工学研究科 知能システム科学専攻
Interdisciplinary Graduate School of Science and Engineering, Tokyo Institute of Technology
nagata@fe.dis.titech.ac.jp

小林 重信
Kobayashi Shigenobu

(同 上)
kobayasi@dis.titech.ac.jp

東条 敏
Tojo Satoshi

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

keywords: memetic algorithm, genetic local search, vehicle routing, local search

Summary

Applications of memetic algorithms (MAs) are usually computationally expensive. In this paper we suggest efficient search limiting strategies for local search used in MAs because local search is the most time consuming part of MAs. The suggested strategies are applied to a recently proposed powerful MA for the capacitated vehicle routing problem (CVRP). Experimental results on the well-known benchmarks show a significant speed-up of 80 % in running time without worsening the solution quality. Moreover, the MA dominates state-of-the-art heuristics for the CVRP with respect to both the computation time and the solution quality.

1. は じ め に

近年、大規模な組合せ最適化問題に対する厳密解法の限界が知られるようになり、メタヒューリスティクスを用いた近似解法の開発が活発に行われている。遺伝的アルゴリズム (Genetic algorithm, GA) もその一つであるが、GA の特徴は複数の解 (集団) による確率的な多点探索と、交叉や突然変異オペレータを用いて新たな解候補を生成することである。特に交叉オペレータにより集団から選択した 2 つ (それ以上でも良い) の解候補を組合わせて新たな解を生成することで、GA が高い大域的探索能力を発揮することが期待されている。一方、GA の局所的探索能力の欠如が指摘されており、近年は GA と局所探索を組み合わせた枠組みである Memetic algorithm (MA) [Moscato 89, 片山 08] の開発が盛んに行われている。同様の枠組みは Genetic local search [Ulder 90] とも呼ばれるが、本論文では MA と呼ぶことにする。

車両配送問題 (Vehicle routing problem, VRP) は物流におけるコスト最適化をモデル化した組合せ最適化問題で、代表的な NP-困難クラスの問題の一つである。特に Capacitated VRP (CVRP) は VRP の最も基本的な問題であり、これまでにさまざまメタヒューリスティクスの枠組みが適用されている。例えば CVRP で良い結果を

出している近似解法として、Guided local search に基づいた手法 [Mester 05, Mester 07a], Large neighborhood search に基づいた手法 [Pisinger 07], Tabu search に基づいた手法 [Rochat 95, Tarantilis 05] などが有名である。一方、MA に基づいた手法 [Prins 04, Alba 04] も提案されているが、前述の手法に較べ得られる解の質で劣る (後述するが計算時間でも劣る)。その原因の一つが、これらの MA では交叉方法として Edge recombination や Order crossover のような解候補の順列表現に対して汎用的に適用可能な単純な方法を用いており、交叉が有効に機能していなかったためであると考えられる。そこで、[Nagata 07a, Nagata 09] では CVRP の性質を考慮した交叉法として Edge assembly crossover (EAX) を考案し、単純な MA の枠組みと組み合わせることで非常に良い近似解を得ることに成功している。

[Nagata 09] で提案された MA は CVRP に対して非常に高い近似性能を持つが、前述の state-of-the-art 近似解法と較べて (MA の能力を十分に引き出すためには) 多くの計算コストを要することが問題点として挙げられる。この原因は一般に MA が多点探索であることに加え、交叉で生成した子個体ごとに局所探索を適用するため、1 回の試行で多くの回数の局所探索を実行する必要がある

ためである．一般に MA は計算コストが高い場合が多く，先に述べた CVRP に対する MA [Prins 04, Alba 04] など多くの計算コストを要している．それゆえ，優れた MA を設計するためには計算コストを（解の精度を悪化させずに）削減することが重要な課題である．

本論文では [Nagata 09] で提案された MA を対象に，計算精度を悪化させることなく計算コストを削減するための方法を提案する．通常，MA の計算コストの大半は局所探索に費やされるため，計算コストの削減法として局所探索の適用回数の削減，あるいは局所探索の効率化が考えられる．本論文では後者の方法として，交叉後に得られた子個体の状況に応じて局所探索の探索領域を制限する方法を提案する．提案法により MA の計算時間を解の精度を悪化させることなく 1/5 程度にまで削減した．その結果，現在 CVRP の最も強力と考えられている近似解法を解精度と計算コストの両面で上回る性能を実現した．

本論文の以下の構成は，まず 2 章で車両配送問題について説明する．3 章では MA で用いる局所探索の制限法を提案する．4 章では提案法を適用する MA [Nagata 09] の概要を述べる．5 章では計算機実験の結果を示し，提案手法の効果を検証する．また他手法との比較も行う．6 章では提案法の挙動分析し，関連研究を含めて考察を述べる．7 章は結論である．

2. 車両配送問題

本章では車両配送問題の定義と，この問題に対する代表的な局所探索近傍を紹介する．

2.1 問題の定義

車両配送問題には制約の種類によってさまざま問題タイプが存在するが，本稿では最も基本的な Capacitated VRP (CVRP) を対象とする．CVRP は以下のように定義される．

$G = (V, E)$ を $n + 1$ 個の頂点からなる完全無向グラフとする．ここで， V と E はそれぞれノード集合と枝集合であり，各枝には距離が与えられている．ノード 0 をデポ，ノード $1, \dots, n$ を顧客の集合とし，各顧客 v に対し荷物量 $q_v (> 0)$ が既知であるとする．CVRP はデポから出発した複数台のトラックが全ての顧客を回って荷物を集配しデポに戻るような配送経路において，トラックの総距離が最も短い（近似解法ではなるべく短い）経路を見つける問題である．ただし，各トラックは共通の容量 Q を持ち，それぞれのトラックが受け持つ顧客の荷物の総量が Q を超えることはできない（容量制約）．容量制約により，配送経路は複数のルート（各トラックの配送経路）からなるが，トラックの数も決定変数であることに注意されたい． Q が無限大の場合 CVRP は巡回セールスマン問題（Traveling salesman problem, TSP）と同一

となる．

CVRP には容量制約に加え，各トラックがデポから出発してからデポに戻るまでの期間（トラックの走行時間 + 顧客のサービスに要する時間の総和で定義）が規定値以下でなければならないという制約（期間制約）を課す場合もある．この問題は DVRP と呼ばれるが本論文では対象としない．

2.2 局所探索近傍

局所探索を設計する際に，どのような探索近傍を用いるかは重要な設計項目である．一般に，なるべく多くの種類の近傍を組合せて用いることで局所解の解精度は向上するが，それにとともに計算時間は増加する．本論文では図 1 に示す VRP に対する最も基本的な 5 種類の近傍を組合わせて用いることにする．この近傍のより詳しい解説については [Kindervater 97] などを参照されたい． $2-opt$ 近傍は現在の解において同一のルート内で 2 本の枝を（全ての可能な方法で）つなぎ変えることで得られる解の集合として定義される．以下同様に， $Or-opt$ は同一のルート内で 1 つの顧客ノードの訪問順を移動する， $2-opt^*$ は異なるルート間で 2 本の枝をつなぎ変える， $Relocation$ は異なるルート間で 1 つの顧客ノードの訪問順を別のルートに移す， $Exchange$ は異なるルート間でそれぞれ 1 つの顧客ノードの訪問順を入れ替えることで定義される近傍である．

3. 局所探索の制限法

本章では MA を効率化するための局所探索制限法を提案する．

3.1 サブ近傍の定義

提案法では，現在の解 σ における顧客ノード v 付近の近傍としてサブ近傍 $\mathcal{N}(v, \sigma)$ を定義する．サブ近傍は基本的には図 1 に示した 5 つの近傍の和集合の部分集合で定義されるが，本論文では説明の都合上，図 2 に示すような別の形式で定義する 4 つの近傍の和集合として定義する．図中，ノード w は v から N_{near} （パラメータ）番目までに近い距離にある顧客ノード全てを選ぶものとし， w は v と同一あるいは別のルートに含まれていても良とする．

これら 4 つの近傍は形式的には以下のように定義される．ここで， $v^- (w^-)$ と $v^+ (w^+)$ はそれぞれノード $v (w)$ に先行あるいは後続するノードを表し（各ルートにおける巡回方向は予め決めておく），顧客ノードでもデポでも良い．これらの 4 つの近傍は遷移後の解に枝 (v, w) が必ず含まれているように定義されていることに注意されたい．ただし， $2-opt(v, \sigma)$ 近傍の 2, 3 番目の遷移に関しただけ w と v は異なるルートになくなくてはならない（デポを含まない部分巡回路が生成されるため）．また， $(1, 1)$ -

図1 VRP の代表的な局所探索近傍 (現在の解を構成するルート数が 2 の場合)

図2 顧客ノード v 付近のサブ近傍の定義. v と w は同一ルート内であっても良い.

Interchange(v, σ) には同一ルート内で 2 つの顧客ノードを交換する遷移が含まれており, 図 1 で紹介した代表的な近傍に含まれていないが, 本論文ではそのまま用いることとした.

2-opt(v, σ)

- (v, v^-) と (w, w^-) を削除, (v, w) と (v^-, w^-) を追加
- (v, v^-) と (w, w^+) を削除, (v, w) と (v^-, w^+) を追加
- (v, v^+) と (w, w^-) を削除, (v, w) と (v^+, w^-) を追加
- (v, v^+) と (w, w^+) を削除, (v, w) と (v^+, w^+) を追加

(1,0)-Interchange(v, σ)

- v を $w-w^-$ 間に挿入, v^- と v^+ を結合
- v を $w-w^+$ 間に挿入, v^- と v^+ を結合

(0,1)-Interchange(v, σ)

- w を $v-v^-$ 間に挿入, w^- と w^+ を結合
- w を $v-v^+$ 間に挿入, w^- と w^+ を結合

(1,1)-Interchange(v, σ)

- v を $w-(w^-)^-$ 間に挿入, w^- を v^--v^+ 間に挿入
- v を $w-(w^+)^+$ 間に挿入, w^+ を v^--v^+ 間に挿入

3.2 局所探索の枠組み

本論文で提案する局所探索の枠組みを手続き Local-Search(σ) (図 3) に示す. この枠組みにおいて $List$ は顧客ノードの部分集合であり, $List$ に含まれている顧客集合で定義されるサブ近傍のみを探索することで局所探索の探索領域を制限する. $List$ は局所探索を開始する前に何らかの戦略に従い初期化するものとする (line 1). 各イテレーション (line 4–12) では, 近傍 $\cup_{v \in List} \mathcal{N}(v, \sigma)$ の中から first 改善 (改善する近傍解が見つかり次第遷移) による山登り法により解を更新する^{*1}.

局所探索において $List$ を終始固定しておく場合は, line 8, 11 の手続き (Update で示されている) を行わない. $List$ を更新する場合, Don't look bit (DLB) 戦略を用いる. DLB 戦略はもともとは TSP に対する 2-opt 近傍を

用いた局所探索を高速化するために考案されたアイデア [Bentley 90] であるが, 提案する局所探索の枠組みに自然に導入することができる. DLB 戦略を用いた場合, あるサブ近傍 $\mathcal{N}(v, \sigma)$ に改善解が存在しなかった場合には, v を $List$ から削除し (line 11), 今後 v により定義されるサブ近傍の探索を行わない. これは今後このサブ近傍に現在の解を改善する解が含まれている可能性が少ないためである. 一方, ある遷移が行われた場合には, その遷移に関連した顧客ノード (変化した枝の両端ノード) 全てを $List$ へ (登録されていない場合には) 登録する (line 8). これは, ある遷移により解が変化することで, 変化を受けた顧客ノード付近の近傍にさらに解を改善する遷移が生じる可能性があるからである.

Procedure Local-Search(σ)

begin

```

1 : Initialize  $List$  using a user-defined strategy;
2 : Set  $List$  in random order;
3 : for  $i := 1$  to  $|List|$  do
4 :    $v := List[i]$ ;
5 :   Search for an improving move in  $\mathcal{N}(v, \sigma)$ ;
6 :   if an improving move  $\sigma'$  is found then
7 :      $\sigma := \sigma'$ ;
8 :     Add the customers associated with the
       improving move to  $List$ ; (Update)
9 :   goto line 2;
10: else
11:   Remove  $v$  from  $List$ ; (Update)
12: end if
13: end for
14: return  $\sigma$ ;
end
```

図3 局所探索の枠組み

*1 ランダムな順で v を選択し, (1,0)-Interchange(v, σ), (1,1)-Interchange(v, σ), (0,1)-Interchange(v, σ), 2-opt(v, σ) の順で探索した. 各近傍では w を v から近い順に選んだ.

図4 両親 A と B から交叉や突然変異等で生成された子 C を構成するする枝の分類.

表1 局所探索の制限法

Strategy	Initialize <i>List</i>	Update
Full search (FUL)	All customers	No
Don't-look bids (DLB)	All customers	Yes
Forbidding moves of common edges (FCE)	All customers Note: Moves changing any common edges are forbidden	No
Limiting moves to new edges (LNE)	A set of customers consisting of endpoints of the new edges	No
LNE with DLB (LNED)	A set of customers consisting of endpoints of the new edges	Yes
Limiting moves to new routes (LNR)	A set of customers consisting of customers in the new routes	No
LNR with DLB (LNRD)	A set of customers consisting of customers in the new routes	Yes

3.3 局所探索の制限戦略

通常, MA では交叉や突然変異などで生成された子個体に対して局所探索が適用されるので, *List* は子個体それぞれに対して初期化するものとする. 提案法の基本的なアイデアは, それらのオペレータで変化を受けた部分(顧客の集合)でリストを初期化することで, 変化を受けた部分にのみ局所探索を制限することである.

同様のアイデアは, TSP に対して反復局所探索法 (Iterated local search, ILS) を適用した強力近似解法としてとして知られる Iterated Lin-Kernighan (ILK) 法 [Johnson 97] で用いられているが, これまでに MA に適用された例はない. ILK では Lin-Kernighan (LK) 法と呼ばれる TSP の強力な局所探索法と double bridge 4-opt と呼ばれる(ある種の条件の下で)ランダムに4本の枝を交換する遷移を逐次的に交互に繰り返すことで探索を行う. double bridge 4-opt による遷移後に得られた順回路で変化を受けなかった部分は良い順回路の性質を保持していると期待されるため, ILK では新たに追加された4本の枝の両端の8都市のみに LK 法手続きの探索開始地点を制限^{*2}することで探索範囲を制限している.

ILS のように1つの解を更新していく場合は遷移(摂動)による解の変化を明確に定義できる. 一方, 交叉(突然変異等を併用しても良い)を用いた場合のように2つの親個体 A と B から子個体 C を生成する場合には, A からの変化と B からの変化に分けて考える必要がある. この場合の解の変化を定義するためにいくつか用語を定義する. 図4に2つの親個体 A と B から交叉等により子個体 C を生成した例を示している. ここで, E_A, E_B, E_C をそれぞれ個体 A, B, C を構成する枝の集合とす

る. さらに, E_C を以下に定義する4つの集合へ分割する.

- E_{CS} : 両親に共通して存在する枝集合
- E_{CN} : 両親のいずれにも存在しない枝集合
- E_{CA} : 親 A にのみ存在する枝の集合
- E_{CB} : 親 B にのみ存在する枝の集合

子個体 C において, 親 A から変化を受けた部分は枝集合 $E_{CB} \cup E_{CN}$ と考えることができる. 同様に親 B から変化を受けた部分は $E_{CA} \cup E_{CN}$ と考えることができるが, 以降では一般性を失うことなく親 A からの変化のみを考えるものとする.

本論文では表1に示す7種類の局所探索制限法を実装するが, *List* の初期化方法として以下の3つの方法を考える.

Full (FUL)

$List := V \setminus \{0\}$ と初期化する. この設定は局所探索を制限しない場合の MA の性能を調べるために用いる.

Limiting moves of new edges (LNE)

List を枝集合 $E_{CB} \cup E_{CN}$ を構成している枝の両端の顧客(デポの場合は無視)で初期化する. 図4の例では *List* は12個の顧客ノードで初期化される.

Limiting moves of new routes (LNR)

List を子個体 C を構成しているルート中, A に存在しない新しいルートに含まれる顧客ノード全で初期化する. 新しいルートとは C を構成するルートにおいて $E_{CB} \cup E_{CN}$ を構成する枝を1つでも含んでいるものを指す. 図4に示した例では C を構成する6つのルートの内3つが新しいルートとなる.

提案法以外の局所探索制限法として, 局所探索中に両親に共通する枝 (E_{CS}) を削除するような遷移を禁止する

*2 通常, 全ての都市を開始地点として LK 法の手続きを行う. その結果, 改善する遷移が見つからなければ局所解となる.

方法を実装する（表 1 中，FCE 戦略）．このアイデアは対象問題のランドスケープが Big Valley 構造を持つ場合に有効と考えられ，同様のアイデアが TSP [Merz 01]，QAP [Merz 97]，さらには CVRP [Kubiak 07] に対する MA に適用されている．この戦略を用いる場合は $List := V \setminus \{0\}$ と初期化し， E_{CS} に属する枝が削除されるような遷移を直接禁止することで探索の制限を行う．

4. Memetic アルゴリズム

本章では提案法を適用する MA [Nagata 09] の概要を示す．

4.1 世代交代モデル

MA の手続きを図 5 に示す． N_{pop} と N_{ch} はパラメータで，それぞれ集団サイズと各両親から生成される子個体の数である． m はルートの個数でこの MA は与えられたルート数のもとでトラックの総走行距離の最小化を行う．まず，ルート数が m である解から構成される初期集団を生成する（line 1）．生成法は 4.4 節で述べる．各世代（line 3–12）では，集団中の個体がちょうど一度ずつ親 p_A としてランダムな順番で選ばれる（line 3, 5）．各両親に対して交叉 EAX を適用して N_{ch} 個の子個体を生成する（line 7）．ただし，EAX は容量制約に違反した実行不可能解を生成するため，修正操作によりこれを実行可能解へ修正する（line 8）．そして，生成されたそれぞれの解に局所探索を適用する（line 9）．生成された N_{ch} 個の子個体の中に親 p_A よりも評価値の良い解が存在すれば，集団中の個体 $x_{r[s]}$ (p_A として選ばれた個体) をその個体で置換える（line 11）．後で述べるように，交叉 EAX（と修正操作）で生成される子個体は p_A と類似する傾向があるため，多様性維持の観点から生成子個体は p_A として選ばれた個体とのみ生存選択を行っている．

本来 m は決定変数であるので，適切な m （例えば，既知最良解のルート数）の値をあらかじめ知ることはできない．そこで，[Nagata 09] では小集団（集団サイズ：10）の MA を 1-3 個の m の候補に対して実行し，最も総走行距離が短くなった m を用いてより大きな集団（集団サイズ：100）で MA を実行する．この前処理により 9 割程度は既知最良解を与える m を得ることができる．

4.2 交 叉 EAX

[Nagata 07a, Nagata 09] で提案された交叉 EAX は TSP に対して提案された交叉 EAX [永田 99] を CVRP に対して拡張した方法である．以下と図 6 に CVRP に対する EAX の手続きを示す．Step1 – Step5 により与えられた 2 つの親個体 p_A と p_B （それぞれ m 個のルートで構成）から m 個のルートで構成される子個体を生成するが，容量制約を満たすことは保証されない．また，Step 3 – Step5 を繰り返すことで複数の子個体を生成するこ

procedure MA(N_{pop}, N_{ch}, m)

begin

1 : 初期集団 $\{x_1, x_2, \dots, x_{N_{pop}}\}$ の生成

2 : **repeat**

3 : $r[]$ を 1 から N_{pop} までのランダムな順列とする;

4 : **for** $i := 1$ to N_{pop} **do**

5 : $p_A := x_{r[i]}$; $p_B := x_{r[i+1]}$;

6 : **for** $j := 1$ to N_{ch} **do**

7 : $c_j := \text{EAX}(p_A, p_B)$;

8 : $c_j := \text{Repair}(c_j)$;

9 : $c_j := \text{Local_Search}(c_j)$;

10: **end for**

11: $x_{r[i]} := \text{Select_Best}(p_A, c_1, c_2, \dots, c_{N_{ch}})$;

12: **end for**

13: **until** 終了条件が満たされる;

14: **return** 集団中の最良個体;

end

図 5 MA の枠組み

とができる．

Step 1 E_A と E_B をそれぞれ両親 p_A と p_B を構成する枝の集合とする． $G_{AB} = (V, E_A \cup E_B \setminus (E_A \cap E_B))$ を定義する．

Step 2 G_{AB} 上の枝を AB -cycle へと分割する．ここで， AB -cycle とは 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．

Step 3 生成された全ての AB -cycle の中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．ここで， E -set は AB -cycle の任意の組合せで構成される枝集合と定義する．

Step 4 E_A から E -set に含まれる E_A に属する枝を取り除き，これに E -set に含まれる E_B に属する枝を付け加えることで中間個体を生成する．この操作により，中間個体は m 個のデボを含むルートと幾つかの（ゼロの場合もある）デボを含まない部分順回路から構成される．

Step 5 中間個体に部分巡回路が含まれる場合は，部分巡回路をランダムな順番で選択し， m 個のルートのどれかに結合することで， m 個のルートからなる子を得る．各部分巡回路の結合は 2-opt^* と類似の操作（部分順回路とルートから 1 本ずつ枝を削除し，2 本の枝を加えて結合）により行われるが，結合した際の総距離が最も小さくなる結合位置を探索する．

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．本論文では以下の 3 つの方法を用いる．

Single 戦略： AB -cycle を任意に 1 つ選択して E -set を構成する．

図6 EAX のアルゴリズム

Block 戦略： まず *AB-cycle* を任意に 1 つ選択し、これを *center AB-cycle* と呼ぶ。次に、*center AB-cycle* と少なくとも 1 点で重なりを持つ全ての *AB-cycle* の中で、*AB-cycle* のサイズ（含まれているノード数）が *center AB-cycle* のサイズ以下であるものを全て選択して *E-set* を構成する。

Random 戦略： 個々の *AB-cycle* を確率 1/2 でランダムに選択して *E-set* を構成する。

Single 戦略を用いた場合（例えば、図 6 中 (a)–(a'')），生成される中間個体は p_A に類似しており（ p_A に対する少数の枝の交換で生成される），Step 5 の後に得られる子個体も同様である。Block 戦略を用いた場合（例えば、図 6 中 (b)–(b'')），生成される中間個体と子個体は Single 戦略を用いた場合と比較すると、 p_A に対するより多くの枝の交換で生成される（ただし、実験的には p_B よりも p_A に近い）。Random 戦略を用いた場合は、中間個体と子個体は平均的には両親の枝を均等に含むように生成される。[Nagata 09] では、MA の探索前半に Single 戦略を用い、後半で Block 戦略を用いる方法を採用しているが、本論文でもこの方法を用いる。一方、Random 戦略は両親として選ばれた 2 つの解を均等に組合わせて子を生成するという意味では自然であるが、Single+Block 戦略に較べて良い結果が得られないために [Nagata 09] では用いられていない。本論文では提案手法の効果を検証するための比較実験として用いる。

4.3 修正操作

交叉 EAX で生成された解が容量制約に違反する場合、修正操作により実行可能解へ修正する。修正操作はペナルティー関数 $F_g(\sigma)$ を用いた局所探索により行われるが、 $F_g(\sigma) = F(\sigma) + \alpha \cdot P_c(\sigma)$ と定義する。ここで、 $F(\sigma)$ はもともとの目的関数であるルートの総距離、 $P_c(\sigma)$ は各ルートの容量制約超過量の和、 α はペナルティー項の重みである。 α の決定法は 4.4 節で述べる。

図 7 に修正操作のアルゴリズムを示す。各イテレーション（line 2–5）では、容量制約に違反しているルートをランダムに選択し（line 2）、選択されたルートを構成している顧客集合で定義されるサブ近傍^{*3}内でペナルティー関数を最小にする解へ遷移する。ただし、ペナルティー項が減少する遷移のみを許可するものとする。イテレーションは実行可能解が得られるか（line 6）、可能な遷移が見つからなくなる（line 5）まで続けられる。実行可能解が得られなかった場合にはその解は捨てるものとする。

4.4 初期集団の生成法

m 個のルートからなる初期解は、 n 個のルートからなる解を初期状態として、ペナルティー関数 $F_g(\sigma)$ を用いた局所探索（山登り法）により生成する（ルート数が減少する遷移を許すものとする）。一般に、パラメータ α の値を小さく設定するほど、この局所探索で得られる局所解のルート数は少なくなるが、 α の値は局所解のルート数が最も m になる場合が多くなるように設定し、この

*3 [Nagata 09] ではサブ近傍の概念は導入されていないが、本論文ではサブ近傍を用いて記述した。

値を修正操作でも用いるものとする．得られたルート数が m の局所解（実行不可能解）に対して修正操作を適用して実行可能解を生成し，この操作を集団サイズ分の初期解が得られるまで繰り返して初期集団を生成した．より詳しい方法については [Nagata 09] を参照されたい．

5. 提案法の評価実験

3章で提案した局所探索制限法を4章で述べた MA に適用してその効果を確認する．さらに，得られた結果を CVRP の state-of-the-art 近似解法で得られる結果と比較する．

5.1 実験設定

3.3節で述べた7種類の局所探索制限法（FUL, DLB, FCE, LNE, LNER, LNR, LNRD）（表1）の効果を調べるために，それぞれの制限法を用いた局所探索（図3）を MA（図5）に適用する．MAのパラメータは基本的には [Nagata 09] と同じで以下のように設定する．集団サイズと子個体生成数は $N_{pop} = 100$ ， $N_{ch} = 30$ とする．サブ近傍に関するパラメータは $N_{near} = 10$ とする．交叉 EAX の設定には以下の2種類を用いる．Single+Block 戦略：始めに Single 戦略のみを用い，50世代にわたり集団中の最良個体の評価値が改善しない場合に Block 戦略へ移行する，さらに50世代にわたり集団中の最良個体の評価値が改善しない場合に MA の終了とする．Random 戦略：Random 戦略を用い，MA が50世代にわたり集団中の最良個体の評価値が改善しない場合終了する．

[Nagata 09] ではルート数 m を決定するための前処理（4.1節参照）を行っているが，局所探索制限法の比較実験では m は既知最良解の値を問題ごとに与えるものとする．これは，前処理で得られる m の違いによる実験結果の差異を排除するためである．ただし他種法との比較を行う場合は，各問題で試行ごとに前処理で m を決定する

Procedure REPAIR(σ)

begin

1 **:repeat**

2 : Randomly select an infeasible route $\rightarrow r$;

3 : Search $\sigma' \in \bigcup_{v \in cust(r)} \mathcal{N}(v, \sigma)$ s.t. $F_g(\sigma')$ is minimized subject to $P_c(\sigma') - P_c(\sigma) < 0$;

4 : **if** σ' exists **then** Update $\sigma := \sigma'$;

5 : **else break**;

6 **:until** σ becomes an feasible solution;

7 **:return** σ ;

end

図7 Algorithm of the repair procedure

ものとする．

それぞれの設定の MA を CVRP の最も代表的な3つのベンチマークセットへ適用する．Christofides の14問題（C1–C14）[Christofides 79] は顧客数が50–199，Taillard の13問題（T75a–T385）[Taillard 93] は顧客数75–385，Golden の20問題（G1–G20）[Golden 98] は顧客数200–483からなる問題である．ただし期間制約が課されている問題（DVRP）と顧客数が100よりも少ない問題は紙面の都合上省略する．各問題に対してそれぞれの設定で10試行を行うものとする．プログラムはC++を用いて記述され，計算機実験は2.4GHz AMD Opteron プロセッサ上で行った．

5.2 局所探索制限法の効果

表2に7種類の局所探索制限法を MA（EAX に Single+Block 戦略あるいは Random 戦略を用いた場合）に適用した結果を示す．ただし，紙面の都合上 EAX に Random 戦略を用いた場合の結果の一部は省略し，Golden のベンチマークに対する結果のみを示す．表3にそれぞれの局所探索制限戦略を，EAX に Single+Block 戦略あるいは Random 戦略を用いた場合のそれぞれについて，FUL 戦略と比較した場合の相対的な解の質と計算時間を示す．具体的な定義は表を参照されたい．これらの結果については Golden のベンチマークだけでなく Christofides と Taillard のベンチマークに対する結果も示す．これらのベンチマークを構成している問題については表4も参照されたい．

まず，Golden のベンチマークに対して EAX に Single+Block 戦略を用いた場合の結果に着目する．表3より，FUL 戦略と比較して残りの6つの戦略は計算時間を削減していることがわかる．ただし，FCE 戦略を用いた場合は計算時間を44%まで削減しているものの，計算精度は悪化している．また，LNE 戦略を用いた場合では計算時間の削減効果は大きく17%となっているが，計算精度は悪化している．一方，LNR 戦略を用いた場合は計算精度をほとんど悪化させることなく，計算時間を23%まで削減している．次に，FUL, LNE, LNR の各戦略に List の更新を行った場合（それぞれ，DLB, LNED, LNRD），それぞれの場合で（更新を行わなかった場合に比較して）計算精度を落とすことなく計算時間をさらに削減している．特に，LNRD 戦略では FUL 戦略と比較して計算精度を悪化させることなく計算時間を19%まで削減している．個々の問題の詳細なデータについては，表2を参照されたい．

次に，Golden のベンチマークに対して EAX に Random 戦略を用いた場合に着目する．表2から分かるように，Random 戦略を用いた場合は Single+Block 戦略を用いた場合よりも得られる解の精度が良くない場合が多い．ただし，問題 G10 と G11 では明らかに Random 戦略を

用いた場合の結果が良く、G19 でもベスト解での比較では Random 戦略の結果が良い。表 3 から分かるように、EAX に Random 戦略を用いた場合は LNE 戦略と LNR 戦略共に計算時間を削減する効果は少ない。

表 3 より、Christfides と Taillard のベンチマークの結果も Golden のベンチマークの結果と同様の傾向を示していることが分かる。ただし、顧客数が比較的大きく難しい問題である Golden のベンチマークで最も大きな差が現れている。特に、Christfides と Taillard のベンチマークのいくつかの問題に対しては EAX を用いた MA で容易に既知最了解へ到達できるため、戦略間の計算精度の差異は小さくなっている。

5.3 他手法との比較

表 4 に EAX に Single+Block 戦略を用いて局所探索制限法に LNRD 戦略を用いた MA (MA (LNRD) と表記) と他の近似解法を比較した結果を示す。他手法としてそれぞれのベンチマークセットで良い性能を実現しているものを選んだ。比較手法の結果は問題ごとに 1 試行のみ行っている場合は得られた解の総走行距離を、複数回の試行を行っている場合は平均した結果を示す。ただし Rochat の結果はさまざまなパラメータ設定で得られた最良解を示す(そのデータしか得られないため)。提案 MA に関しては 10 試行中の最良解と平均した結果、さらに 1 試行当たりの平均計算時間(秒)を示す。各表の下段にはベンチマークセットごとの平均的な結果として、既知最良解からの誤差 (Dev. (%))、平均計算時間 (Ave. CPU (min))、使用計算機 (Computer) を示す^{*4}。

表 4 の MA (LNRD) の結果は表 2 の LNRD の結果と異なっているが、この理由はルート数 m を決定する前処理の有無によるものである(表 4 の MA ではルート数 m を決定する前処理あり)。表 4 中、Nagata で示した手法 [Nagata 09] は基本的には本論文で用いた MA に FUL 戦略を適用した場合に等しい。ただし、[Nagata 09] では局所探索の過程で解が親 A と等しくなった場合に局所探索を停止する方法を導入することで計算時間を削減しているため、単純に FUL 戦略を適用した場合に比べ 1-2 割程度計算時間が削減されていることに注意されたい。

まず Golden のベンチマークでの結果を見る。Nagata らの MA [Nagata 09] が提案されるまでは Mester2 で示した Guided local search に基づく手法 [Mester 07a] が CVRP に対する最も優れた近似解法であった。Nagata らの MA は解の精度では Mester2 よりも良い結果を実現しており、表 4 中 Golden のベンチマークの既知最良解の全てを発見している。しかし、Mester2 [Mester 07a] や Pisinger らの Large neighborhood search [Pisinger 07] などに比較して多くの計算時間を要している。一方、MA

(LNRD) は計算時間で Mester2 と比較して 2.5 倍(計算機の違いを考慮して)程度にまで接近している。ただし、Mester2 では各問題に対して既知最良解のルート数を与えており、さらに計算時間として最良解が発見された時点での計算時間を計測している。同様の実験設定を用いた場合、MA (LNRD) の Ave. CPU time は 5.6 分、Dev. は 10 試行の平均で 0.12% であった。また、参考のため集団サイズを 50 にして MA (LNRD) を実行し、Mester2 と同様の実験設定を用いた場合、MA (LNRD) の Ave. CPU time は 2.7 分、Dev. は 10 試行の平均で 0.19% であった。よって、MA (LNRD) は解の質と計算時間の両面で Mester2 の結果を上回る性能を示した。

Christofides のベンチマークにおいても、Goldedn のベンチマークの結果と同様に、MA (LNRD) の結果が解の質と計算コストの両面でほぼ従来手法を上回っている。ただし、Christofides の問題では Mester2 で得られる解の質が最も良いが、計算時間は MA (LNRD) より多い。これは、Mester2 の実験設定では問題 C5 で総走行距離が 1291.29 の解を発見するために 2160 秒という長い計算時間をかけているためである。Taillard のベンチマークでは MA (LNRD) が比較手法に較べ明らかに良い結果を得ている。ただし、Mester2 の方法がこのベンチマークには適用されていないことに注意されたい。

6. 局所探索制限法の挙動・考察

本章では表 2 で結果を示した MA における局所探索制限法の効果を分析し、関連研究を含めて考察する。

図 8 に Golden のベンチマークの 1 問題である G9 ($n = 255$, $m = 14$) に表 2 で結果を示した MA (1 試行) それぞれにおける局所探索の挙動を示す。図 8 中、(a) は交叉 EAX と修正操作後に得られる子個体の E_{CS} , E_{CN} , E_{CA} , E_{CB} (3.3 節参照) の構成比を累積グラフで示している。(b) は FUL, LNE, LNR それぞれの戦略を用いた場合の *List* を構成する顧客ノード数を示している。ただし、*List* が全ての顧客ノードで構成されている場合を 1 と規格化した。(c) は局所探索当 1 回あたりの解の評価回数を示している。ただし、各戦略を同一の条件下で比較するため、EAX に Single+Block 戦略を用いた MA に局所探索制限なし (FUL) の実験で得られた各世代の集団 (130 世代目に EAX は Single 戦略から Block 戦略に切り替わっている) に対して交叉、修正操作、それぞれの局所探索を適用してこれらのデータを取るものとする。挙動を表すデータは各世代での平均値で示すものとし、10 世代ごとに表示した。

まず、EAX に Single+Block 戦略を用いた場合を考察する。図 8(a) に示すように $E_{CN} + E_{CB}$ の割合は 3-6% と小さく、子 C は親 A からの比較的局所的な変化で生成されている。これは EAX に Single あるいは Block 戦略

^{*4} 提案 MA を Opteron 2.4GHz マシンと Pentium-4 2.8GHz マシンで実行した結果、実行時間の比は約 1.0:1.4 であった。

表 2 各局所探索制限法を用いた場合の MA の探索性能 (Golden らのベンチマーク [Golden 98] における結果) . 最初の列は問題名, 顧客数, 既知最良解 (ルート数/総走行距離) を示している. 結果は各問題に対して, それぞれの手法ごとに 10 試行で得られた最良解の総走行距離 (best), 平均値 (average), 1 試行当たりの平均計算時間 (time) で示されている. 最後の 3 行に全ての問題で平均した結果として, 最良解の既知最良解からの誤差 (Dev. of best), 平均値の既知最良解からの誤差 (Dev. of aver.), 平均計算時間 (Ave. CPU) を示す.

Problem Instance		EAX(Single + Block)							EAX(Rand)		
		FUL	DLB	FCE	LNE	LNED	LNR	LNRD	FUL	LNE	LNR
G9 (255) 14/580.42	best	581.04	580.57	580.96	580.87	580.91	581.05	579.99	580.35	580.33	580.35
	average	581.55	581.33	582.38	582.00	582.03	582.03	581.84	580.87	581.04	581.02
	time (s)	1088	869	472	289	236	402	308	1593	1276	1592
G10 (323) 16/738.49	best	739.06	739.06	740.18	739.28	738.93	739.03	738.75	737.33	737.33	737.18
	average	739.97	740.10	741.21	740.43	740.46	740.07	740.08	739.40	739.21	737.74
	time (s)	1736	1385	831	377	341	595	456	2962	2288	3203
G11 (399) 18/914.72	best	915.51	913.89	916.11	915.07	914.80	914.76	914.17	912.93	913.60	912.79
	average	916.77	915.83	917.22	916.65	916.66	916.31	916.71	920.23	925.85	921.18
	time (s)	2375	2120	1297	548	567	888	625	3898	2465	3718
G12 (483) 19/1106.76	best	1105.78	1107.58	1107.52	1107.37	1107.58	1105.34	1107.32	1149.64	1149.67	1151.63
	average	1107.74	1108.84	1110.03	1109.01	1109.80	1108.21	1108.61	1152.36	1155.42	1153.88
	time (s)	4304	2956	1892	908	800	1282	1149	865	666	670
G13 (252) 26/857.19	best	857.19	857.19	857.19	857.19	858.19	857.19	857.19	857.19	857.19	858.14
	average	858.47	858.38	859.26	858.75	859.54	858.84	857.91	858.23	858.41	859.19
	time (s)	1218	1106	411	223	173	288	222	1702	1209	1281
G14 (320) 30/1080.55	best	1080.55	1080.55	1080.55	1080.55	1080.55	1080.55	1080.55	1080.55	1080.55	1080.55
	average	1080.55	1080.84	1080.96	1081.11	1080.99	1080.55	1080.55	1080.62	1080.62	1080.55
	time (s)	1503	1166	625	325	234	346	257	1528	1019	1244
G15 (396) 33/1340.24	best	1342.02	1342.17	1342.68	1343.19	1339.26	1342.95	1341.46	1376.88	1375.89	1377.17
	average	1342.87	1344.03	1344.53	1344.96	1343.19	1343.61	1343.58	1381.23	1380.90	1382.85
	time (s)	2719	1738	962	390	335	477	445	754	938	732
G16 (480) 37/1619.93	best	1619.77	1614.58	1621.07	1621.31	1620.69	1619.31	1619.09	1650.20	1665.34	1661.50
	average	1622.44	1620.65	1623.57	1622.77	1622.25	1622.07	1622.59	1668.31	1671.50	1670.54
	time (s)	3276	3093	1596	524	475	698	539	1495	1199	1479
G17 (240) 22/707.76	best	707.76	707.76	707.76	707.76	707.76	707.76	707.76	707.76	707.76	707.76
	average	707.77	707.79	707.77	707.77	707.78	707.77	707.78	707.81	707.79	707.79
	time (s)	782	633	330	87	78	109	98	424	293	325
G18 (300) 27/995.13	best	995.39	995.39	995.13	995.62	995.44	995.13	995.16	1008.35	998.73	1001.44
	average	996.38	995.81	996.23	996.20	996.64	996.08	995.98	1020.57	1021.88	1018.28
	time (s)	1518	1296	546	185	157	200	202	293	287	334
G19 (360) 33/1365.97	best	1366.31	1366.17	1366.05	1366.88	1366.24	1366.54	1366.38	1365.99	1366.00	1365.63
	average	1366.69	1366.65	1367.10	1367.49	1367.13	1367.05	1366.86	1366.55	1373.30	1373.19
	time (s)	1925	1598	919	197	192	247	212	1438	842	862
G20 (420) 38/1820.02	best	1821.19	1820.18	1822.21	1822.76	1821.47	1821.27	1819.38	1881.09	1884.42	1881.22
	average	1821.91	1822.12	1823.72	1824.09	1822.57	1822.55	1821.19	1886.81	1888.16	1886.99
	time (s)	3009	2715	1493	344	357	420	401	285	288	322
Dev. of best. (%)		0.035	-0.004	0.077	0.072	0.037	0.027	-0.0018	1.066	1.078	1.088
Dev. of aver. (%)		0.124	0.118	0.208	0.178	0.173	0.140	0.1283	1.448	1.597	1.503
Ave. CPU (s)		2121.6	1723.4	948.3	367.0	329	496.6	410.0	1437.0	1064.6	1314.0

表 3 各問題グループにおける FUL 戦略との相対的な比較結果. $Rel. dev. (ave.) = \frac{100}{|Prob|} \cdot \sum_{i \in Prob} \frac{aver_i^A - aver_i^{FUL}}{aver_i^{FUL}}$ と定義する. ここで, $Prob$ は問題集合, $aver_i^A$ は表 2 における問題 i の手法 A の結果 (average) を表している. 同様に, $Rel. dev. (best)$ と $Rel. dev. (time)$ も定義するが, 後者は FUL 戦略の結果を 100 % として表示する.

	EAX(Single + Block)							EAX(Rand)		
	FUL	DLB	FCE	LNE	LNED	LNR	LNRD	FUL	LNE	LNR
Golden										
Rel. dev. (best) (%)	0.000	-0.039	0.042	0.036	0.002	-0.007	-0.036	0.000	0.012	0.021
Rel. dev. (ave.) (%)	0.000	-0.006	0.084	0.055	0.050	0.016	0.006	0.000	0.147	0.055
Rel. time (%)	100.0	82.0	43.7	17.3	15.2	23.2	18.8	100.0	80.5	91.5
Christofides										
Rel. dev. (best) (%)	0.000	-0.022	0.000	0.024	-0.022	-0.022	-0.022	0.000	0.003	0.000
Rel. dev. (ave.) (%)	0.000	-0.006	0.010	0.013	-0.001	-0.003	-0.008	0.000	-0.002	-0.004
Rel. time (%)	100.0	88.3	33.0	26.1	23.6	36.0	29.4	100.0	38.3	53.3
Taillard										
Rel. dev. (best) (%)	0.000	-0.002	-0.004	0.003	-0.004	-0.004	-0.003	0.000	0.001	-0.003
Rel. dev. (ave.) (%)	0.000	0.001	0.001	0.002	0.002	-0.001	-0.002	0.000	0.032	-0.024
Rel. time (%)	100.0	74.8	36.9	30.1	25.3	40.9	30.6	100.0	43.6	62.5

表 4 他手法との比較．比較手法は著者名で表しているが，それぞれ以下の文献で提案された手法である：Tarantilis [Tarantilis 05]，Prins [Prins 04]，Mester1 [Mester 05]，Mester2 [Mester 07a]，Mester3 [Mester 07b]，Pisinger [Pisinger 07]，Nagata [Nagata 09]，Rochat [Rochat 95]，Alba [Alba 04]．

Christofides		Tarantilis	Prins	Mester2	Pisinger	Nagata	MA (LNRD)		
Problem	Best-known	1 run	1 run	1 run	Ave. of 10 runs	Ave. of 10 runs	Best of 10 runs	Aave. of 10 runs	time (sec)
C3 (100)	826.14	826.14	826.14	826.14	826.99	826.14	826.14	826.14	9
C4 (150)	1028.42	1028.42	1031.63	1028.42	1034.20	1028.42	1028.42	1028.42	26
C5 (199)	1291.29	1311.48	1300.23	1291.29	1306.63	1291.84	1291.45	1291.45	72
C11 (120)	1042.11	1042.11	1042.11	1042.11	1042.46	1042.11	1042.11	1042.11	13
C12 (100)	819.56	819.56	819.56	819.56	819.56	819.56	819.56	819.56	5
Dev. (%)		0.31	0.20	0.000	0.38	0.009	0.002	0.002	
Ave. CPU (min)		5.4	5.0	7.2	2.3	1.4	0.4 × 10	0.4	
Computer		Pentium 400	Penti-3 1.0GHz	Penti-4 2.8GHz	Penti-4 3.0GHz	Opteron 2.4GHz	Opteron 2.4GHz	Opteron 2.4GHz	

Taillard		Rochat	Mester1	Alba	Mester3	Nagata	MA (LNRD)		
Problem	Best-known	best of different settings	1 run	1 run	1 run	Ave. of 10 runs	Best of 10 runs	Ave. of 10 runs	time (sec)
T100a	2041.34	2047.90	2041.34	2047.90	2041.34	2041.34	2041.34	2041.34	15
T100b	1939.90	1940.61	1939.90	1940.36	1939.90	1940.26	1939.90	1940.12	14
T100c	1406.20	1407.44	1406.20	1411.66	1406.20	1406.20	1406.20	1406.20	13
T100d	1580.46	1581.25	1581.25	1584.20	1581.25	1580.46	1580.46	1580.46	18
T150a	3055.23	3070.91	3055.23	3056.41	3055.23	3055.23	3055.23	3055.23	20
T150b	2727.20	2733.60	2727.67	2732.75	2727.77	2728.02	2727.20	2727.89	62
T150c	2341.84	2364.31	2343.11	2364.08	2361.56	2358.94	2358.66	2358.95	70
T150d	2645.39	2663.20	2645.40	2654.69	2645.40	2645.39	2645.39	2646.60	35
T385	24372.27	24435.50	24855.32	25015.01	24965.01	24386.84	24372.51	24393.74	543
Dev. (%)		0.35	0.23	0.57	0.37	0.09	0.07	0.10	
Ave. CPU (min)		N/A	28.8	N/A	35.4	6.3	1.5 × 10	1.5	
Computer		SGI 100	Penti-4 2.0GHz	Penti-4 2.8GHz	Penti-3 800MHz	Opteron 2.4GHz	Opteron 2.4GHz	Opteron 2.4GHz	

Golden		Tarantilis	Prins	Pisinger	Mester2	Nagata	MA (LNRD)		
Problem	Best-known	1 run	1 run	Ave. of 10 runs	1 run	Ave. of 10 runs	Best of 10 runs	Ave. of 10 runs	time (sec)
G9 (255)	580.42	585.43	596.89	590.33	583.39	581.46	580.99	581.93	325
G10 (323)	738.49	746.56	751.41	751.36	741.56	739.56	739.04	740.23	461
G11 (399)	914.72	923.17	939.74	926.57	918.45	916.27	914.40	916.17	924
G12 (483)	1106.76	1130.40	1152.88	1125.22	1107.19	1108.21	1107.21	1108.81	1121
G13 (252)	857.19	865.01	877.71	874.24	859.11	858.42	857.19	858.53	249
G14 (320)	1080.55	1086.07	1089.93	1103.53	1081.31	1080.84	1080.55	1080.60	252
G15 (396)	1340.24	1353.91	1371.61	1366.23	1345.23	1344.32	1342.54	1344.22	421
G16 (480)	1619.93	1634.74	1650.94	1645.67	1622.69	1622.26	1619.34	1621.64	631
G17 (240)	707.76	708.74	717.09	710.59	707.79	707.78	707.76	707.76	126
G18 (300)	995.13	1006.90	1018.74	1007.84	998.73	995.91	995.13	996.04	242
G19 (360)	1365.97	1371.01	1385.60	1377.88	1366.86	1366.70	1366.37	1366.83	261
G20 (420)	1820.02	1837.67	1846.55	1834.70	1820.09	1821.65	1820.46	1822.34	549
Dev. (%)		0.92	2.13	1.45	0.22	0.12	0.03	0.14	
Ave. CPU (min)		45.5	65.5	8.4	4.5	30.8	7.8 × 10	7.8	
Computer		Pentium 400	Penti-3 1.0GHz	Penti-4 3.0GHz	Penti-4 2.8GHz	Opteron 2.4GHz	Opteron 2.4GHz	Opteron 2.4GHz

図 8 局所探索制限法の挙動．Single+Block 戦略を用いた場合（上段），Random 戦略を用いた場合（下段）．

を用いた場合，子個体は親 A からの比較的局所的な変化で生成されるためである．それゆえ，(b) に示されるように LNE 戦略と LNR 戦略のそれぞれで List を構成する顧客の数が FUL 戦略の場合と比較して大きく削減される．(c) と (b) の結果において FUL, LNE, LNR の各戦略を比較すると，List のサイズが大きいほど局所探索での評価回数が多くなっていることが分かる．また，Don't look bits 戦略による List の更新を追加した場合（DLB, LNER, LNRD），さらに評価回数を削減していることが分かる．次に，(c) と表 3 の結果（G9 の計算時間）を比較すると，局所探索での評価回数が少ないほど MA の実行にかかる計算時間も削減されていることが分かるが，MA では局所探索に大半の計算時間が費やされることから妥当な結果といえる．ただし，FCE 戦略の結果は例外で，評価回数の割に計算時間が大きい．この原因は，解遷移により両親に共通する枝が削除されるかどうかを判定するための計算コストがかかるためである．

次に EAX に Random 戦略を用いた場合を考察する．この場合，図 8(d) に示されているように $E_{CN} + E_{CB}$ の割合は 20–40% と Single+Block 戦略を用いた場合と比較して大きい．それゆえ，(e) に示されるように，LNE 戦略と特に LNR 戦略において List のサイズを削減する効果も少なくなる．その結果，(c) と (f) の比較から分かるように，局所探索制限戦略に LNR 戦略あるいは LNE 戦略を用いた場合の局所探索に要する解の評価回数は，EAX に Single+Block 戦略を用いた場合に較べて多く，MA 全体の計算時間も多くなる（表 2）．

既に述べたように，LNE や LNR 戦略（さらに LNER, LNRD 戦略）で局所探索の探索範囲が大幅に削減されるためには EAX の Single+Block 戦略のような局所的な交

叉を用いることが必要である．局所的な交叉のもとでのアイデアは TSP のための交叉 EAX の工夫として提案され，交叉を局所化することにより GA で得られる解の質が向上すること [永田 06]，さらに交叉の計算コストを削減できること [Nagata 07b] が示されている．Nagata らの MA [Nagata 09] で交叉 EAX を CVRP に拡張した際も，MA で得られる解精度を向上させる目的で EAX に Single 戦略や Block 戦略のような局所的な交叉を用いており，実際に Random 戦略のような大域的な交叉を用いた場合よりも良い解精度を実現している（表 2）．

本論文では，MA で局所的な交叉を用いた場合には，局所的な交叉で変更を受けた部分に局所探索の探索範囲を制限することで効果的に MA の計算コストを削減できることを示した．提案法の基本的な考え方は他の組合せ最適化問題にも適用することができるが，どのように局所探索の探索範囲を制限すれば良いかは問題に応じて適切に設計する必要がある．例えば，CVRP に対して本論文で示した例では，LNE 戦略を用いた場合は探索範囲を絞り込み過ぎたために解の質が悪化した．一方，CVRP の特徴を考慮した LNR 戦略を用いた場合には解精度を悪化させることなく探索範囲を削減することに成功した．

7. 結 論

本論文では先に CVRP に対して提案されていた優れた探索精度をもつ MA に対し，MA の計算コストを削減するための局所探索制限法を提案した．提案法では交叉等のオペレータで生成された子個体と一方の親個体（類似している方）との変化部分に局所探索の探索領域を制限することで，探索精度を落とすことなく計算コストを削減

できることを示した。この結果、提案した MA は CVRP に対する state-of-the-art 近似解法を計算精度と計算コストの両面で上回る性能を実現した。一般に MA は計算時間が多くかかることが問題点の一つであるが、本論文で提案した局所探索制限法の枠組みを応用して、さまざまな組合せ最適化問題に対して計算精度と計算コストの両面ですぐれた MA を考案したいと考えている。

◇ 参 考 文 献 ◇

- [Alba 04] E. Alba, B. Dorronsoro: Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms, Proc. of the 4th Int. Conf. on Evolutionary Computation in Combinatorial Optimization, LNCS 3004, pp. 11–20 (2004).
- [Bentley 90] J. L. Bentley: Experiments on Traveling Salesman Geuristics, Proc. of the First Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 91–99 (1990).
- [Christofides 79] N. Christofides, A. Mingozzi, P. Toth: The vehicle routing problem, Combinatorial optimization, N. Christofides, A. Mingozzi, P. Toth and C. Sandi (eds), Wiley, Chichester, pp. 315–38 (1979).
- [Golden 98] B.L. Golden, E.A. Wasil, J.P. Kelly, I.M. Chao: Metaheuristics in vehicle routing, Fleet management and logistics, T.G. Crainic and G. Laporte (eds), Kluwer, Boston, pp. 33–56 (1998).
- [Johnson 97] D.S. Johnson, L.A. McGeoch: The traveling salesman problem: a case study, in Local Search in Combinatorial Optimization, E. Aarts and J.K. Lenstra (eds.), Jhon Wiley & Sons, pp. 215–310 (1997).
- [片山 08] 片山 謙吾, 石淵 久生: Memetic アルゴリズム, 計測と制御, Vol. 47, No. 6, pp. 487–492 (2008).
- [Kindervater 97] G.A.P. Kindervater, M.W.P. Savelsbergh: Vehicle routing: handling edge exchanges, in Local Search in Combinatorial Optimization, E. Aarts and J.K. Lenstra (eds.), Jhon Wiley & Sons, pp. 337–360 (1997).
- [Kubiak 07] M. Kubiak, P. Wesolek: Accelerating Local Search in a Memetic Algorithm for the CVRP, Proc. of the 7th Int. Conf. on Evolutionary Computation in Combinatorial Optimization, LNCS 4446, pp. 142–153 (2007).
- [Merz 97] P. Merz, B. Freisleben: A genetic local search approach to the quadratic assignment problem, Proc. of the 7th Int. conf. on Genetic Algorithms, pp. 238–245 (1997).
- [Merz 01] P. Merz, B. Freisleben: Memetic Algorithms for the Traveling Salesman Problem, Complex Systems, Vol. 13, pp. 297–345 (2001).
- [Mester 05] D. Mester, O. Bräysy: Active guided evolution strategies for large scale vehicle routing problems with time windows, Computers & Operations Research, Vol. 32, No. 10, pp. 1593–1614 (2005).
- [Mester 07a] D. Mester, O. Bräysy: Active guided evolution strategies for large scale capacitated vehicle routing problems, Computers & Operations Research, Vol. 34, No. 10, pp. 2964–2975 (2007).
- [Mester 07b] D. Mester, O. Bräysy, W. Dullaert: A multi-parametric evolution strategies algorithm for vehicle routing problems, Expert Systems with Applications, Vol. 32, No. 2, pp. 508–517 (2007).
- [Moscato 89] P. Moscato: On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms, C3P Report 826, California Institute of Technology (1989).
- [Nagata 07a] Y. Nagata: Edge Assembly Crossover for the Capacitated Vehicle Routing Problem, Proc. of the 7th Int. Conf. on Evolutionary Computation in Combinatorial Optimization (2007), LNCS 4446, pp. 142–153.
- [Nagata 09] Y. Nagata, Olli Bräysy: Edge Assembly based Memetic Algorithm for the Capacitated Vehicle Routing Problem, Networks, Vol. 54, No. 4, pp. 205–215 (2009).
- [永田 99] 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).
- [永田 06] 永田 裕一: 局所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, 人工知能学会誌, Vol. 21, No. 2, pp. 195–204 (2006).

- [Nagata 07b] 永田 裕一: 局所的な交叉 EAX を用いた GA の高速化と TSP への適用, 人工知能学会論文誌, Vol. 22, No. 5, pp. 524–552 (2007).
- [Pisinger 07] D. Pisinger, S. Ropke: A general heuristic for vehicle routing problems, Computers & Operations Research, Vol. 34, No. 8, pp. 2403–2435 (2007).
- [Prins 04] C. Prins: A simple and effective evolutionary algorithm for the vehicle routing problem, Computers & Operations Research, Vol. 31, No. 12, pp. 1985–2002 (2004).
- [Rochat 95] Y. Rochat, E. Taillard: Probabilistic diversification and intensification in local search for vehicle routing, Journal of Heuristics, Vol. 1, pp. 147–167 (2005).
- [Taillard 93] E. Taillard: Parallel iterative search methods for vehicle routing problems, Networks, Vol. 23, No. 8, pp. 661–673 (1993).
- [Tarantilis 05] C.D. Tarantilis: Solving the vehicle routing problem with adaptive memory programming methodology, Computers & Operations Research, Vol. 32, No. 9, pp. 2309–2327 (2005).
- [Ulder 90] N.L.J. Ulder, E.H.L. Aarts, H.-J. Bandelt, P. J. M. van Laarhoven, E. Pesch: Genetic Local Search Algorithms for the Travelling Salesman Problem, Proc. of the 1st Int. Conf. on Parallel Problem Solving from Nature, LNCS 496, pp. 109–116 (1990).

〔担当委員: 村田 忠彦〕

2009 年 7 月 31 日 受理

—— 著 者 紹 介 ——

永田 裕一 (正会員)

1994 年東京工業大学応用物理学科卒業, 2000 年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。2000 年同大学院総合理工学研究科リサーチアシリエイト, 2001 年北陸先端科学技術大学院大学情報科学専攻助手, 2009 年より東京工業大学大学院総合理工学研究科知能システム科学専攻助教, 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。

小林 重信 (正会員)

1969 年 3 月東京工業大学工学部応用化学科卒業, 1974 年 3 月同大学理工学研究科経営工学専攻博士課程修了。工学博士。1974 年 4 月同大学工学部制御工学科助手, 1981 年 8 月同大学総合理工学研究科システム科学専攻助教, 1990 年 8 月同研究科生命科学専攻教授, 1996 年 5 月より同研究科知能システム科学専攻教授, 現在に至る。創発システム論, 進化計算, 強化学習などの研究に従事。

東条 敏 (正会員)

1981 年東京大学工学部計数工学科卒業, 1983 年東京大学大学院工学系研究科修士課程修了。博士 (工学)。1983–1995 年三菱総合研究所, 1995 年北陸先端科学技術大学院大学情報科学研究科助教授, 2000 年同教授。自然言語の形式意味論および人工知能の論理の研究に従事。情報処理学会, 人工知能学会, ソフトウェア科学会, 言語処理学会, 認知科学会各会員。