Vol.2011-MPS-82 No.25
2011/3/8

物流最適化に向けたGAとバックトラックの融合によるTSP近似解法

櫻井義尚† 塚本奈津貴† 高田考平† 鶴田節夫†

配送ルートなどの最適化システムでは，数十～数百拠点(都市)以上の最短道順探索問題（巡回セールスマン問題：TSP)を対話的応答時間（3 秒程度以下）かつ専門家程度の誤差（3%前後以下）で解かなければならない．しかも，シンプルで分かりやすい方法が望まれる．ヒューリスティクスが複雑だとユーザに難しいと嫌われ実用にならない上，開発者，特に顧客担当の技術者も変化の多い現場や配送条件に合わせた修正を容易にしたいためである．これらの要求を満たすために，最近挿入法(NI)や 2-opt など基本的な方策（発見的知識）やメタ戦略（バックトラック＆ランダムリスタート法 : BR と略す）を遺伝的アルゴリズム GA に組込むシンプルで分かりやすい方法を提案する．提案方式で 1000 回試行した結果，500 都市前後までの TSPLIB の問題を 3 秒以内に最悪誤差が 3%以下で, 解くことができた．本方式は，多忙な現場の人になじみのある分かり易いヒューリスティックスである NI や 2-opt を部分的に適用した突然変異を中心とする GA に BR を融合したシンプルなものであり，上記の柔軟な対応が容易である．

An Approximation Algorithm based on Backtrack & GA to Optimize Delivery Schedule

YOSHITAKA SAKURAI† NATSUKI TSUKAMOTO†
KOUHEI TAKADA† SETSUO TSURUTA†

A delivery route optimization that improves the efficiency of real time delivery or a distribution network requires solving several tens to hundreds but less than 2 thousands cities Traveling Salesman Problems (TSP) within interactive response time, with expert-level accuracy (less than about 3% of error rate). Moreover, as for the algorithms, understandability and flexibility are necessary because field experts and field engineers can understand and adjust it to satisfy the field conditions. To meet these requirements, a Backtrack and Restart Genetic Algorithm (Br-GA) is proposed. This method combines Backtracking and GA having simple heuristics such as 2-opt and NI (Nearest Insertion) so that, in case of stagflation, GA can restarts with the state of populations going back to the state in the generation before stagflation. Comparison based on the experimental results and consideration proved that the method meets the above requirements more than other methods judging from not only optimality but also simplicity, flexibility, and expandability in order for this method to be practically used.

1. 序論

小包や手紙の配送から複数の企業がからむ製品供給/分配のための配送まで，最適化を要する多くの物流ネットワークがある．これらの効率を高めるためには，配送ルートつまり多数の配送先への配送順序を最適化することが必要である．1 回の配送では，数十箇所～数百箇所以上の異なった場所を回る．したがって，このような配達ルートの最適化は，中大規模の巡回セールスマン問題（Traveling Salesman Problems, TSP）[1][2]としてモデル化できる．TSP は計算複雑性理論において，NP 困難と呼ばれる問題のクラスに属する組み合わせ最適化問題である．n-都市の TSP を解くためには n!オーダーの組み合わせを処理する必要があり，計算量の爆発を引き起こす．

一方，実際は，定式化が困難な社会的人間的条件がからむため，現場での実用には，求解結果を人間が確認する必要がある．すなわち，近似解が実用できるか人間のユーザが即座にチェックし，時には手動修正したり，代替案を選択したりしなくてはならない．つまり実用上，TSP の解法には人間の介入をスムーズに行うための対話（リアルタイム）応答性が要求される．

また，領域の専門家によって生成された近似解は，数学上の最適解と比較すると，3%程度の誤差が存在する．しかし，それ以上に精度の悪い解を出して，実用的な問題を引き起こすようなことはない．つまり，解の精度としては，厳密解が必要なわけではないが，3%程度以下の誤差の範囲内での解が必要になる．実際の配送などの利用現場では，誤差の大きい近似解を出力するとユーザからの信頼を失い，実用されなくなる．従来の近似 TSP 解法[1][2] は，この点に問題があり，誤差の保証ができない．

さらに，回り易い慣れたルートなど配送者の都合も含めた現場特有の条件に合わせたルートの出力が必要となる．つまり，最適化といっても最短時間，最短距離だけでなく，各配送サイトや配送関係者さらには地域・社会の特殊な要求や事情に依存する面も多い．このため，現場の SE(システムエンジニア)が現場の専門家などと相談しながら方式(アルゴリズムやプログラム)やパラメータの調整および出力結果の対話的修正が必要なことも多い．そこで，彼ら現場の専門家が通常用いるなじみ深くシンプルで現場の SE にも分かりやすい方策(ヒューリスティクス)で求解する必要がある．

そこで，上記の NI などのようなシンプルで分かり易く，なじみ易いヒューリスティクスを組み込んだ幾つかのタイプの遺伝的アルゴリズム（Genetic Algorithm, GA）を提案してきた[4]-[6]．しかし，約 200 箇所(都市)を超える規模の問題（TSP）になると，解の精度が 4%を超える場合が多くなった．

本論文ではこれを解決するために，GA やバックトラック＆リスタート(BR)などメタヒューリスティクスの効率的融合活用方式を提案する．GA で進化が停滞気味にな

† 東京電機大学 情報環境学部 情報環境学科
Department of Information Environment, School of Information Environment, Tokyo Denki University

ると，ランダムリスタートを適用した後，選択・淘汰操作の前に上記の NI や 2-opt などのシンプルなヒューリスティクスを用いて改良する保護育成により集団の多様性を保つ方式である．

以下，第 2 節では，配送ルート最適化問題とその技術課題を説明する．第 3 節で，関連手法について，第 4 節で，問題を解決するための方法を提案する． 第 5 節では，実験と結果を示し，実験結果に基づいて解法の有効性を実証する．最後に第 6 節では，結果を結論づける．

2. 関連研究

2.1 配送ルート最適化問題の定式化

物流ネットワークの配送ルート最適化問題を定式化すると以下のようになる：配送網は重み付き完全グラフ $G=(V,E,w)$で表現する．ここで V はノード（頂点）集合であり，ノード v_i （i=1,…,N）は巡回する配送元や配送先つまり巡回拠点あるいは都市を表す．N はノード数である．E は枝集合であり，枝 e_{ij} はノード v_i とノード v_j を結ぶルートを表す．w は枝の重み集合であり，枝の重み d_{ij} はノード v_i とノード v_j の２地点間距離である．このようなグラフ $G=(V,E,w)$上で最も短い巡回路長をもつハミルトン閉路(2)を求める問題を巡回セールスマン問題（Traveling Salesman Problem, TSP），このような閉路を最適解と呼ぶ．ルートの適応度は総距離で評価する．総距離が短いものほど適応度が高くなる．このように配送ルート最適化問題を TSP として定式化する．

2.2 実用に向けた TSP 近似解法の関連研究

上記のように，実用に向けた**配送ルート最適化問題で**は，対話時間内に，専門家レベルの精度だけでなく社会的事情も絡む現場特有の条件に合ったしかも回り易い解が要求される．現場ユーザのやり方に近く，説得容易で受け入れられやすい方法，分り易く現場の要求に合わせてカスタマイズし易い方法でないと高速･高精度なだけでは，実用化が困難である．

そこで，上記の NI などのようなシンプルで分かり易く，なじみ易いヒューリスティクスを組み込んだ幾つかのタイプの遺伝的アルゴリズム（Genetic Algorithm, GA）を提案してきた[4]-[6]．例えば，2-opt 型 GA は一部の問題パターンでは解の精度が良いが，別のパターンでは局所最適解に陥る傾向がある．そこで，最近挿入法（Nearest Insertion, NI）を使ったブロック NI 型 GA をカスケードさせることにより 2-opt 型 GA の短所を補う外部多世界 GA（Mow-GA，Multi-outer-world GA）方式[4]を提案した．しかしながら，Mow-GA は前段の GA で生成された優秀な個体の形質が子孫に遺伝しない．つまり，初期個体からランダムに再スタート（ランダムリスタ―ト）する．また，3 秒程度の求解時間では，問題規模が大きいと各 GA で十分な世代数を確保出来ない．さらに，探索過程における各ヒューリスティクスの相補作用は望めない[5]．そこで，ヒューリスティクスの相補作用による探索効率を高めた内部多世界 GA（Multi-inner-world Genetic Algorithm, Miw-GA）[6]を，次に提案した．これは，相互に弱点を補う関係にある 2 つのヒューリスティクス，1 つは 2-opt 型の突然変異，もう 1 つはブロック NI 型突然変異を 1 世代内に実行する方式である．単純で分かりやすい求解過程にするため，GA の操作も突然変異に限定した．この方法はさまざまな問題パターンに対し，対話時間内に高精度な解を得た．しかし，約 200 箇所(都市)を超える規模の問題（TSP）になると，解の精度が 4%を超えてしまった．

そこで，内部ランダムリスタート GA（Inner Random Restart Genetic Algorithm, Irr-GA）[23]を提案した．これは，個体集団の多様性を保つ為に，ランダムリスタートを用いたものであった．この手法は，600 拠点以下の TSP に対しては，3 秒以内に 3%以下の精度を実現できたが，それ以上の規模の TSP に対しては最適化性能にまだ問題があった．

2.3 TSP 解法に関する関連研究

配送の最適化を図る手法は数多く存在するが，物流業務の実情を踏まえることが困難なため，実際に使用するには問題が多い[22]．

近似解法は様々な手法が提案されている．TSP のヒューリスティックスサーチ手法としては，Lin-Kernighan (LK)法が有名である．LK 法や，その改良方式[2][8]，あるいは Karp の分割アルゴリズム[3]等は，得られる近似解の最適性は高いが複雑である．

特に近似解法では，世界トップクラスの性能を持つ LKH[7]は初期処理を含め 1000 都市までなら 3 秒程度以内で，2000 都市前後までなら約 10 秒以内で解ける．しかし，処理が複雑なため解法が分かりにくくユーザに受け容れ難い上，現場特有の条件への柔軟な対応が容易でない．これらのヒューリスティックスは SA（Simulated Annealing: 焼きなまし法）や TS（Tabu Search:タブーサーチ），GA などのメタヒューリスティックスサーチ手法と組み合わせて使われることが多い．

SA[9]は理論的には，局所解に陥る危険性を少なくして，最悪誤差が 3%以下の解を得ることができる．しかし，焼きなまし用の冷却速度やコスト関数など SA の多数のパラメータを問題に合わせて設定することは非常に難しく，また，上記の準最適解を求めるには多大の計算時間を要するため，物流や配送の現場では使い難い．

TS[10]も実用的に最適解に近い解を求めるためには計算時間が長い．また，局所探索としての特性が強く，解の多様化能力が弱いなどの欠点が指摘されているが，その改良手法が金澤らにより提案されている[11]．

いわゆるランダムリスタート法[12]は，ランダムな初期解を改善するために 2-opt 法などのローカルサーチを適用し，近似最適解を得ることができる．具体例として，貪欲ランダム化適応型探索手法（GRASP：Greedy Randomized Adaptive Search

Procedure）[13]や繰返し数の制限により応答性保証を可能とした改良型ランダムリスタート法[14]がある．この改良型ランダムリスタート法は， 40 都市の TSP を約 100 ミリ秒 3%以内の誤差で解いたが，200 拠点を越えると精度の保証が困難な場合が生じた．

TSP を効率的に解く為の GA についても様々な方式が提案されている．たとえば，LK 法や Edge Assembly Crossover （EAX）[15]，Distance-preserving crossover （DPX）[16]などを利用することにより 1000 都市から 10000 都市にも及ぶ超大規模 TSP を効率的に解くための GA 手法が提案されている[17][18]．EAX や DPX は親ツアーの枝の特質を厳密に子ツアーへと継承させることができる．しかし，これらの交叉オペレータはツアーの解析が必要なため，計算時間が長く，超大規模な TSP 以外に用いるのは効率的でない場合がある．文献[17]は 2 種類の方法を多くのケースで比較している．それは，300-10000 都市 TSP には Cga-LK が有利であるが，198 都市 TSP にはランダム-LK が有利であることを示している．したがって，数千都市の TSP を効率よく解ければ, 1—2 桁低い巡回拠点数や都市数の TSP が必ず効率よく解けるというわけではない．

本研究が対象とするようなスケール（数十から数百都市以上，2 千都市未満）の TSP に関して，文献[17]では，TSPLIB の問題 lin105 を解いている．また，文献[19]でも，この問題を解く上での様々な交叉オペレータの性能比較実験が行われている．また，本研究が対象とするスケールの TSP を高速に解くための GA 手法としては，Yan らによる GA 手法[20]が提案されている．

文献[5][6]において，前提案方式 Mow-GA，Miw-GA と Random-LK，文献[19]での実験の最良交叉オペレータを用いた GA 及び金澤らによる TS や Yan らによる GA との比較実験を行い，提案手法の速度と精度における優位性を実証した．

3. バックトラック＆リスタート GA（Br-GA）

3.1 提案手法のコンセプト

本論文では, バックトラック＆リスタート GA（Backtrack & Restart Genetic Algorithm, Br-GA）を提案する．Br-GA は，メタヒューリスティクスとしての GA とバックトラック＆リスタートメカニズムを組み合わせ，そこに複数種類のシンプルなヒューリスティクス（発見的知識 : 後述の NI と 2-opt）を組み込んでいる．GA による解探索は，初期個体と呼ばれる解候補の複数生成とそれらの改良・評価・選択（淘汰）による生き残り解候補選択を 1 世代の処理として, これを数世代繰り返すことにより行われる．GA による解探索を効率的に進めるためには，比較的優良かつ類似性の低い個体を生成することと，生成した個体集団の多様性の維持が重要になる．

Br-GA では，最近挿入法（NI : Nearest Insertion method）や 2-opt などのシンプルな複数のヒューリスティクスを融合する．まず，各遺伝子に対し拠点番号をランダムに割当てる，あるいはそれに NI を適用して初期個体を作る．次に各世代では，親個体の遺伝子の一部に NI を突然変異操作として作用させ，親個体の遺伝子配列情報を部分的に継承しているが一部は異なる子個体を作り，それを 2-opt で局所改良し，集団を進化させる．しかしながら，親以上の評価値を持つ子個体が，規定世代数以上連続して作成できなかった時，つまり進化が停滞したと思われる時，指定世代において，最良個体あるいは評価値が規定値以下の子個体を進化が停滞する前の状態に戻すもしくはランダムに初期化し直すバックトラック＆リスタート（BR）と呼ばれるメタ戦略を適用する．また，生成された子個体のうち同じ評価値を持つなど類似した個体にも BR を行う．

次に，上記のヒューリスティクスを適用し，これらの新生個体が生存できる様に育成・保護しながら世代としての最終評価を行い，親個体も含めた中から集団サイズ分の個体を選択（淘汰）する．

こうして，世代の途中に全く新規の個体群を生成するとともに，複数のヒューリスティクスの適用によって，様々な方向から局所的改良を行う．あるいは，それらがすぐに淘汰されないように保護・育成の期間や手段を講じて，集団の多様性を保ちながら局所収束をできるだけ避け求解する方式である．

3.2 個体の生成と異種方策の交互利用による局所収束の防止

本提案の GA では，各個体（染色体）の各遺伝子は TSP におけるノード番号（配送先の識別番号）を表す．従って染色体は巡回順序すなわち，配送ルートを表すノード列である．

GA の初期段階では，ランダムに解の候補になる個体（集団）のグループ（初期個体集団）が作られる．但し，この場合も，同じ点を 2 回以上通るなど明らかに効率の悪いルート（致死遺伝子）が生成されないように，例えば最近挿入法（Nearest Insertion method, NI）を使用して始祖個体を生成するランダム NI 生成を用いる．また，その結果に対し，本例では 2-opt を適用し局所最適化した後，初期の親個体とする．こうして，対話リアルタイムな応答性能を目指し，最適化効率を高める．

(1) ランダム NI による始祖個体生成

局所最適解への収束を避けるには，初期個体のランダム性が重要である．しかし，完全にランダムでは，収束速度は遅い．したがって，NI 方式にランダム性を加えたランダム NI 方式を工夫した．即ち，ランダムな順序にノードを並べ，その順に NI(最近挿入)法[2]を使用して，それらをツアー（最初は空の部分ツアー）に挿入し，初期解としてのツアーを構成する．NI 方法の擬似コードを**エラー! 参照元が見つかりません。**に，ツアー長 n を入力として，生成されたツアー$tour_{after}$を出力するランダム NI の擬似コードを図 3 に示す．

```
Algorithm NI ( tour_before{x1, x2, ... , xn}, x_new)
  L_before ← length of tour {x1, x_new, x2, ..., xn}; // x1 の後に x_new を経由する時のツアー長
  Insersion location i ← 1;
  For j ← 2 to n Do
    L_after ← length of tour {x1, ... , xj, x_new, ... }; // xj の後に x_new を経由する時のツアー長
    If L_after < L_before Then  L_before ← L_after;  Insersion location i ← j;  End If
  End For
  insert node x_new between x_i and x_{i+1} of tour_before{x1, x2, ... , xn};
End /*NI*/
```

図 2　NI 法

Figure 2　Algorithm of NI method

```
Algorithm random NI (作成するツアーの長さ n)
  tour_before ← random tour {x1, x2, ..., xn} ;   /* 順序のランダムなツアーを作る */
  tour_after ← empty tour { } ;   /* 空ツアーを作る */
  /* ランダムツアーのノードを先頭から順に NI(Fig.2 参照) */
  For i ← 1 to n Do tour_after ← NI(tour_after, x_i in tour_before); End For
  return tour_after; /* 生成されたツアーを出力 */
End /*random NI*/
```

図 3 ランダム NI 法

Figure 3　Algorithm of random NI method

```
Algorithm 2-opt mutation(親個体 tour_par {x1, x2, ..., xn} )
list ← empty set {} ;/* 除外リストを空にする */
i ← select random number from 1 to n ; /* ノード x_i をランダムに選択*/
add {i , i+1} to list; /* 除外リストに x_i を加える*/
L_par ← length of tour_par; L_chi ← MAX;
Do /* 改善されるまで組み替え続ける */
  /*既選択ノードを除いた別のノード x_j をランダムに選択*/
  j ← select random number from 1 to n except list ;
  tour_chi ← tour_par {x1, ..., xi,xi+1, ... , xj,xj+1,..., xn}の部分ツアー{xi+1, ... , xj}を反転 ;
  /* tour_chi {x1,..., xi,xj,xj-1, ... , xi+2,xi+1,xj+1,..., xn}を生成
    （図 5 のように，ツアー中の二つのリンクを取り出して，
      その二つのリンクを交換するのと同じ意味）*/
  L_chi ← length of tour_chi ;
Until L_par >L_chi OR list is full{1, 2, … , n}
return tour_chi;/* 2-opt 突然変異操作をした子個体を出力 */
End /* 2-opt mutation */
```

図 4　2-opt 型突然変異

Figure 4　Algorithm of 2-opt mutation

3.3 シンプルなヒューリスティクスによる個体の改良

改良方策はシンプルで現場の配送者に分かり易い NI や 2-opt を用いる．NI は現場の人間に馴染み易い上，削除・挿入が前後の拠点との通路に影響を与えるだけのシンプルで汎用性の高い（例えば一方通行のある非対称 TSP への適用が容易な）方策だからである．但し，これらの方策は，基本的に巡回点数の 2 乗オーダの計算負荷なので，対話リアルタイム性確保のため，NI では適用対象にする遺伝子を一部のブロックに制限し(ブロック NI)，2-opt もある一定限度までしか改良しない．また，各巡回点の近傍をあらかじめ計算しておき，これを考慮した置き換えを行う．これらも多忙な現場の熟練者になじみのあるやり方で，説明時に納得され受け入れられやすいと考える．

(1) 2-opt 型突然変異

この方法は，2-opt[2] のような単純な局所探索ヒューリスティック法と GA の突然変異オペレータを結合することによって収束速度の改良を図る．親個体 $tour_{par}$ を入力として，子個体 $tour_{chi}$ を出力する 2-opt 型突然変異の擬似コードをに示す．

(2) ブロック(NI)型突然変異

2-opt 型突然変異は効率的にツアーを改良でき，短時間に良好な解を得ることができる．しかし，このような効率的な探索は，逆に局所最適に陥る危険性を高めてしまう．より最適性の高い解を得るためには，局所最適から脱出するためにツアーの一部を破壊し再構築する操作が必要である．このため 1 つの遺伝子だけでなく，その近傍の遺伝子も再構築するブロック NI 型突然変異を提案する．親個体 $tour_{par}$ を入力として，子個体 $tour_{chi}$ を出力するブロック NI 型突然変異の擬似コードを図 6 に示す．

```
Algorithm block-type mutation (親個体 tour_par {x1, x2, ..., xn} )
tour_chi ← copy tour_par ;   /*親個体のコピーを子個体とする*/
i ← select random number from 1 to n ; /* ノード x_i をランダムに選択*/
/* 近傍の大きさ r は問題依存知識により決定 */
r ← 0 から B ×（デポと呼ぶ配送基地から最も遠いノードまでの距離）の間の乱数
tour_sub ← ノード x_i とその近傍のノード{x_{i-r}, ... , x_{i+r}};
tour_chi から{x_{i-r}, ... , x_{i+r}}を削除;
/* 抜き出したサブツアーを先頭から順に NI で挿入 */
For j ← i - r to i + r Do
  tour_chi ← NI(tour_chi , x_j in tour_sub );
End For
return tour_chi;/*ブロック型突然変異操作をした子個体を出力 */
End /* block-type mutation */
```

図 6　ブロック NI 型突然変異

Figure 6　Algorithm of block-type mutation

3.3.1 バックトラック＆リスタートによる個体集団の多様性の維持

最良個体が数世代にわたって更新されなかった場合，現在の集団で進化停滞が発生したと見なし，進化停滞発生より前世代の状態に集団の状態をバックトラックして(戻して）リスタートする．このように，進化停滞にある集団を進化停滞前のような特定のチェックポイントまで巻き戻す処理を backtrack and restart (BR)処理と呼ぶ．この BR 処理により集団の多様性を保つ事ができる．

BR 処理は GA が対象とする TSP の規模（ノード数）により異なる．大規模 TSP を解く場合，BR 処理は進化停滞したエリート個体を停滞前の状態に戻し，2-opt 型突然変異により改良される（進化計算がやり直される）．これに対して，小規模 TSP を解

く場合，BR 処理は停滞したエリート個体を新生個体に置き換える．つまり，ランダムに個体を生成し，その育成を行う．

```
/* ランダム NI とローカル探索による個体生成*/
Algorithm randomStart()
  tour ← ランダム NI による生成個体;
  tour を 2-opt 突然変異によりを改良
  tour の適応値を評価
  Return tour;
End /* randomStart */
/*ブロック型，2-opt 型突然変異による子個体生成*/
Algorithm makeChild(parent /*parent tour*/)
  child ← parent からブロック型突然変異で子個体生成
  child を 2-opt 突然変異によりを改良
  child の適応値を評価
  Return child;
End /* makeChild */
Algorithm Br-GA()
size_pop ←集団サイズ; size_chi ←子個体集団サイズ; size_tsp ← 対象とする TSP のノード数;
time_now ←現在実行時間; time_inter ←現在実行時間 - time_now ; time_end ← 終了時間;
For i ← 1 to size_pop Do /*初期個体集団の生成*/
  survival_i ← randomStart (); /*生存集団の i 番目個体*/
End For
gen_now ← 0; /*現在の世代数*/
While time_now + time_inter < time_end Do
  親個体集団 parent ←生存集団 survival; gen_now ←gen_now + 1; /*第 gen_now 世代*/
  For j ← 1 to size_chi Do /*カスケード処理による子個体生成*/
    親 parent_p をランダム選択; child_j ← makeChild( parent_p );
  End For
  子個体集団を適応度の高い順に並べ替え;
  If 最良個体の更新があった Then
    gen_back ← gen_now; /*バックトラックする世代数を記録*/
    tour_back ← 更新された最良個体;
  Else If 最良個体の更新が一定世代無い Then
  /* バックトラッキング処理 */
    If size_tsp >1000 Then/* 大規模 TSP の場合 */
    gen_lag ← gen_now - gen_back ; tour_sub ← tour_back ;
    Else gen_lag ← gen_now ; tour_sub ← randomStart ();/* ランダムリスタート個体 */ End If
    /*新生個体の育成処理*/
    For j ← 1 to gen_lag× size_chi Do
      tour_sub をブロック型突然変異によりを改良; tour_sub を 2-opt 突然変異によりを改良;
      If 子のエリート個体 child_0 < tour_sub Then child_0 ← tour_sub; End If
      If 親のエリート個体 parent_0 < tour_sub Then parent_0 ←tour_sub; break;/*ループ終*/End If
    End For
  End If
  p ← 1; c ← 1;
  For j ← 1 to size_pop Do /*親子集団の適応値の高い上位 size_pop 個が生存*/
    IF parent_p > child_c Then survival_j ← parent_p; p ← p + 1;
    Else survival_j ← child_c ; c ← c + 1; End If
  End For
  time_inter ← 現在実行時間 - time_now ; time_now ← 現在実行時間;
End While; End /* Algorithm Br-GA() */
```

図 7 提案方式(バックトラック＆リスタート GA：Br-GA)の擬似コード

Figure 7 Pseudo code of Backtrack & Restart GA（Br-GA）

BR のタイミングは，一定世代毎あるいは進化が停滞した時とする．例えば，親個体より優秀な子個体が生成されないことが一定世代続いた時とする．同じ評価値を持つ子個体が作成された時にも多様性維持のために BR を行う．

3.4 個体の評価・淘汰と新生個体の保護育成

GA の収束を速め，かつ，高精度な解を得るために，生存個体は親集団と子集団の両方から選択する．バックトラック＆リスタート処理（BR）により生成されたばかりの個体は他の親個体に比べて適応度が低いため，そのまま競争させるとすぐに淘汰されてしまう可能性が高く，多様性の維持に役立たない．そこで，生成後に保護育成を行う事により，他の進化した個体と対等に競争できる機会を与える．世代数が進むほど，生存個体と新しく生成した個体との適応度の差は大きくなる．そのため，世代数に比例して保護育成の期間，保護育成のための改良の回数を多くする．また，保護育成において新しく生成した個体の適応度が親のエリート個体，つまりは現集団での最良個体より高くなった場合は保護育成処理を終了する．これは，少なくともその新生個体は，次の世代で保護なしに生存できるからである．

3.5 提案手法の詳細

上記の手法により構成されるバックトラック＆リスタート GA（Br-GA）の詳細アルゴリズムの基本部分を図 7に示す．実行前に設定すべきパラメータとしては，集団サイズ m，子集団サイズ n，終了時間 $time_{end}$ などがある．本論文の実験では，対話リアルタイム応答を実現するため，終了時間は 3 秒と設定している．また，的確に設定した終了時間内に終了させるため，各世代終了時に次世代の終了予測時間を計算している．

次節では，筆者らの以前の提案方式(Mow-GA, Miw-GA, Irr-GA)と今回の提案方式 Br-GA を比較する．さらに，TSP の近似解法では，世界トップクラスの性能を持つ LKH[7], および厳密解法で世界最高速レベルの Concorde [21]と本論文の提案方式を比較する．厳密解法も最悪誤差が 0 という点において，本論文の前提条件を満たす解法だからである．

4. 実験と結果

4.1 実験

本節では，TSP のベンチマークテストである TSPLIB を用いて，以前の提案解法（Mow-GA, Miw-GA, Irr-GA），いずれも性能・精度において現在の最高レベルと考えられる厳密解法の ABCC（Concorde として有名），近似解法の LKH と本提案解法である Br-GA との比較実験を行う．

実験では，AMD Athlon 64 X2 3800+ 2GHz プロセッサ(シングルコアで起動したため，

Athlon 64 3200+ 2GHz とほぼ同性能） と 1GB のメモリを装備している PC を使用し，C 言語によりプログラムを作成した．コンパイルには Microsoft Visual C++ .NET 2003 ver. 7.1.3091 を使用し，コンパイルオプションは/O2 （実行速度優先） に設定し，Windows XP Professional 上で実行した．

また，GA のパラメータについては, 集団サイズは Mow-GA が 50, Miw-GA が 100, Irr-GA と Br-GA が 30, 突然変異率は Mow-GA と Br-GA が 50%, Irr-GA が 40%, Miw-GA の各突然変異オペレータの選択確率は，ブロック型が 60%，2-opt 型が 40%である．試行回数は各問題 1000 回とした. これらのパラメータは事前の各々数回の試行結果に基づいて優良な値に設定した．Mow-GA，Miw-GA，Irr-GA，Br-GA の終了条件は，3 秒以内となる様に設定した．

4.2 結果

表 1 TSPLIB による関連研究との比較結果
Table 1 Results compared with related works on TSPLIB

TSP 問題名	3 秒実行時の最適解からの誤差の最悪値[%]			
	Mow-GA	Miw-GA	Irr-GA	Br-GA
st70	0.194	0.592	0.98	0.46
eil76	0.788	1.858	0.19	0.18
kroA100	0	0	1.46	1.55
pr107	0	0.189	0.90	1.00
pr136	0.105	1.002	0.00	0.00
pr144	0.008	0.090	0.01	0.00
pr152	0.076	0.184	1.60	1.39
pr226	2.767	0.004	2.25	2.48
a280	14.036	4.187	1.66	1.10
lin318	9.331	3.117	1.81	1.93
pr439	11.535	4.603	2.47	2.94
rat575	9.139	7.027	2.50	2.88

エラー! 参照元が見つかりません。は既提案方式である Mow-GA, Miw-GA, Irr-GA と本論文の提案方式 Br-GA を 12 種類のベンチマーク問題（st70, eil76, kroA100, pr107, pr136, pr144, pr152, pr226, a280, lin318, pr439, rat575）に適用して得た比較結果である．試行回数はいずれも 1000 回，各回の実行時間はほぼ 3 秒である．

TSP の名前（例えば，st70）に含まれる数字（例えば，70）は，都市（巡回拠点）の数を示す．各データ欄は，3 秒以内の対話リアルタイム応答時間で得られた解の全試行中の最悪ケースの誤差率（%）を示す．誤差は，求められた解と厳密解における差を厳密解で割ることにより得られる相対誤差である．厳密解は TSPLIB に公開されている各問題の最短経路長を利用した．

エラー! 参照元が見つかりません。によると，a280 から rat575 までの 200 拠点以上の比較的規模の大きい TSP において，Mow-GA や Miw-GA では，3 秒以内（対話リアルタイム応答時間）での最大誤差率が 4%を越えた．これに対し，Br-GA, Irr-GA では，目標とする最大誤差率 3%以下（2.5%前後）を達成した．200 以下の拠点数の問題では，いずれも 2%以下の誤差で大差はなかった．

実験結果によると，Br-GA と Irr-GA は幅広い問題において最大誤差率を低く抑える事ができた．特に Br-GA は，問題による精度の偏りが少なかった．これはバックトラック&リスタート処理による多様性の維持が有効に働いた結果であると推測される．また，Br-GA と Irr-GA は比較的大きい数百拠点の TSP に対しても最大誤差率を低く抑える事ができた．これは，ブロック NI 型突然変異による局所収束からの脱出と 2-opt 突然変異による局所探索のカスケード処理が比較的大きな問題に対しても有効に働き，無駄な探索が押さえられた結果であると推測される．

表 2 TSPLIB を用いた Concorde および LKH との比較結果
Table 2 Results compared with LKH and Concorde on TSPLIB

TSP 問題名	3 秒実行時の最適解からの誤差の最悪値[%]		ABCC 求解時間 [sec]	LKH 求解時間 pre+max[sec]
	Irr-GA	Br-GA		
pr136	1.13	0.46	4	0.2
pr152	0.19	0.18	8	1.1
rat195	1.59	1.55	22	0.8
kroA200	1.06	1.00	7	0.2
ts225	0.00	0.00	21	0.2
pr226	0.01	0.00	4	0.3
gil262	2.15	1.39	13	0.9
a280	2.02	2.48	38	0.2
pr299	0.81	1.10	39	1.1
lin318	2.27	1.93	10	1.2
pr439	0.78	2.94	216	1.3
rat575	2.16	2.88	363	1.4
rat783	6.36	3.68	38	1.0
u1432	9.63	5.97	3740(rl1323)	5.4
rl1889	ー	10.26	2223(vm1748)	6.1

Pre:枝狩り用の前処理時間，max:10 回の最大実行時間，
AMD Athlon 64 3200+2GHz 換算

次に，表 2 に性能・精度において最高レベルと考えられる厳密解法の ABCC

（Concorde として有名），近似解法の LKH，および本論文の提案方式 Irr-GA との比較結果を示す．第 2 列には，最左列に名前のある TSPLIB の問題に対し，Irr-GA の各種方式を 1000 回試行した時の最大誤差率（%）を示す．中間列は，ABCC（厳密解法なので誤差は 0）の求解時間（秒）である（文献(21)より引用）．CPU は異なるが，a280 などに対し本実験環境で実測した場合，求解時間に一割前後の違いしかなかったので，表 2 にそのままの数値を掲載した．結果を見ると，Concorde では 10 秒以上かかる 200 拠点前後以上の規模の TSP を，Br-GA と Irr-GA は 3 秒以内に 3%程度以下の誤差で解けることを確認することができた．また，1000 拠点前後になると最悪誤差率が Irr-GA は 10%近くになっているが，Br-GA は 5%程度に抑えられている．500 拠点程度以下では Br-GA と Irr-GA の性能に大きな差は見られなかったが，それ以上の場合，Irr-GA の方が最適化精度が圧倒的に良くなっている．

一方，TSP の近似解法で世界トップクラスの性能を持つ LKH は，表 2 の最右列に示すとおり初期処理を含め 1000 都市までなら 3 秒程度以内で，2000 都市前後までなら約 10 秒以内で解ける．もちろん誤差率も 3%以下である．しかし，提案方式である Br-GA でも，800 都市前後までの TSPLIB の問題を 3 秒以内かつ最悪誤差が 3%以下で解くことができた．一方，LKH 法は処理が複雑で分かりにくく，現場特有の条件への柔軟な対応が容易でない．具体的には以下に述べるとおりである．

この種の解法はツールとしてユーザにブラックボックスで提供されるとしても，無条件に受け入れられるわけではない．実際の経験では，現場の責任者や専門家はその仕組みや使われるノウハウを必ず聴く．彼ら自身のやり方と違う場合，特に，複雑で分かりにくいあるいは納得できない場合は受け入れられない．また，現場毎に明示的／暗黙的に特別な条件が絡む場合が多い．この場合，例えば単純に，回る順が絶対ではないが存在する部分が数箇所存在する時など，常に複数の候補を保持する，すなわち多点探索型の GA などのメタヒューリスティクスを使う本方式のほうが現場での簡単な改良には有利である．一般的に LKH 法など巧緻をつくした複雑な方法は柔軟な修正が容易でなく，小回りが利かない．現場のシステムエンジニアや情報技術者が十分に理解し尽すことが困難で，修正時のリスクが高く，柔軟な対応やサービスができない．あるいはコストや時間がかかり，不利である．

本方式では，多忙な現場の人になじみのある分かり易いヒューリスティックスである NI や 2-opt を局所探索用の部品として，シンプルなメタヒューリスティックスである GA や BR などに組み込んだ．また，これらの選択や構成をパラメータで制御可能とした．こうして，多点(複数解による)・大局(一時的な解の悪化を許容する)探索を可能にしたため，探索領域も広く，理解性・柔軟性に優れる．これらは，実用性の高い配送スケジューリングシステムの構築を容易にすると考える．

5. 結論

小包や手紙の配送では，数十～数百拠点（最大 2 千前後まで）の最短道順探索問題（巡回セールスマン問題：TSP)を対話的応答時間（約 3 秒以下）かつ誤差 3%以下で解く必要がある．しかも，シンプルで分かりやすい方法が望まれる．ヒューリスティクスが複雑だとユーザに難しいと嫌われ実用が困難となる．これらの要求を満たすために，メタヒューリスティクスである遺伝的アルゴリズム GA とバックトラック＆リスタート法（BR）を GA に組み込む形で融合し，これに最近挿入法(NI)や 2-opt などのヒューリスティクスを組込んだシンプルで分かりやすい方法 Br-GA を提案した．

実験の結果，提案手法は，500 都市前後までの TSPLIB の問題を 3 秒以内に最悪誤差が 3%以下で，解くことができ，1500 都市前後でも 5%程度以下の誤差であった．また, 世界トップクラスの近似解法 LKH と比較しても最適化性能に大きな差は無く，提案手法の方が柔軟な対応や拡張が容易なため，実用向きである事を示した．

今後は，上記の多様な問題パターンに対しても必要な応答性・最適性を保証するため，分かり易く拡張性の高い本方式の特徴を生かし，GA や BR などのメタヒューリスティクスと NI や 2-opt などのシンプルなヒューリスティクスのよりシンプルで効率的な統合・協調方式を研究する．特に，拠点数が千前後でも 3%以下の誤差の解を 3 秒前後で探索できるような改良方式を研究する．また，一方通行などのある，より実用的・一般的な配送道順の最適化問題（非対称 TSP）についても研究を行う．このため，メタヒューリスティクスやヒューリスティクスの選択及びその組み合わせ統合用パラメータの自動調整方式・学習方式についても研究を進める．

参考文献

1) Lawer, E., Lenstra, J., Rinnony, K., and Shmoys, D.:“The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization”, John Wiley Sons, (1985)
2) 山本芳嗣, 久保幹雄：「巡回セールスマン問題への招待」, 朝倉書店, (1997)
3) R. M. Karp: Probabilistic analysis of partitioning algorithm for the TSP in the plane, Mathematics of Operations Research 2, pp.209-224, 1977.
4) 櫻井義尚, 小野山隆, 久保田仙, 鶴田節夫：「配送問題を対話的時間で実用レベル最適化する多段知能型 GA」, 日本知能情報ファジィ学会誌, Vol.20, No.4 pp.639-652 (2008.08)
5) Sakurai, Y., Onoyama, T., Kubota, S., Nakamura, Y., Tsuruta, S.:”A Multi-world Intelligent Genetic Algorithm to Interactively Optimize Large-scale TSP", Proc. of The 2006 IEEE International Conference on Information Reuse and Integration (IEEE IRI2006), Hawaii, USA, pp248-255 (2006)
6) Y. Sakurai, T. Onoyama, S. Kubota, S. Tsuruta: "A Multi-inner-world Genetic Algorithm to Optimize Delivery Problem with Interactive-time", Proc. of 4th IEEE Conference on Automation Science and Engineering (CASE 2008), Washington DC, USA, pp.583-590, (2008.8).
7) K. Helsgaun, "General k-opt submoves for the Lin–Kernighan TSP heuristic," Mathematical

Programming Computation, vol. 1, pp. 119-163, 2009.
8) Lin, S. and Kernighan, B.W.:“An effective heuristic algorithm for the traveling salesman problem”, Operations Research, Vol. 21, No.2, pp.498-516 (1972)
9) Miki, M., Hiroyasu, T, Jitta, T.:“Adaptive Simulated Anealing for maximum temperature”, Systems, Man and Cyernetics, 2003. IEEE International Conference, vol.1, pp.0-25 (2003)
10) Fang, Y., Liu, G., He, Y., Qiu, Y.:“Tabu search algorithm based on insertion method”, Proceedings of the 2003 International Conference on Neural Networks and Signal Processing, pp.420-423 (2003)
11) 金澤貴彦， 安田恵一郎：「Proximate Optimality Principle に基づく Tabu Search」， 電気学会論文誌. C，電子・情報・システム部門誌， Vol.124，No.3 pp.912-920 (2004). Kanazawa, T., Yasuda, K.,“Proximate Optimality Principle Based Tabu Search”, IEEJ Transactions on Electronics, Information and Systems, Vol.124, No.3, pp.912-920, 2004.
12) 柳浦 睦憲, 茨木 俊秀：「組合せ最適化問題に対するメタ戦略について」 ， 電子情報通信学会論文誌. D-I， 情報・システム， I-情報処理 Vol. J85-D-II，No8 pp.3-25 (2000). Yanagiura, M. and Ibaraki, T., “On Metaheuristic Algorithms for Combinatorial Optimization Problems”, Transactions of the IEICE, Vol.J85-D-II, No.8, pp.3-25, 2000
13) Feo, T.A., Recende, M.G.C., and Smith S. H.:“A Greedy Randomized Adaptive Search Procedure for Maximum Independent Set”, Operations Research, Vol. 42, No.5, pp.860-878 (1994)
14) Kubota, S., Onoyama, T., Oyanagi, K. and Tsuruta, S.:“Traveling Salesman Problem Solving Method fit for Interactive Repetitive Simulation of Large-scale Distribution Network”, Proc. IEEE SMC’99, pp.533-538 (1999)
15) 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). Nagata, Y. and Kobayashi, S., “The Proposal and Evaluation of a Crossover for Traveling Salesman Problems: Edge Assembly Crossover”, Journal of Japan Society for AI, Vol.14, No.5, pp.848-859, 1999.
16) Whiteley, D. and Starkweather, T.:“Scheduling Problem and Traveling Salesman: The Genetic Edge Recombination Operator”, Proceeding of ICGA'89, pp.133-140 (1989)
17) Baraglia, R., Hidalgo, J.I., Perego, R.:“A hybrid heuristic for the traveling salesman problem”, IEEE Transactions on Evolutionary Computation, Vol.5, Issue 6, pp.613-622 (2001)
18) Nguyen, H.D., Yoshihara, I., Yamamori, K., Yasunaga, M.: “Implementation of an Effective Hybrid GA for Large-Scale Traveling Salesman Problems”, IEEE Transactions on Systems, Man and Cybernetics, Part.B, Vol.37, Issue. 1, pp.92-99 (2007)
19) Cheng, C., Lee, W., Wong, K.:“A genetic algorithm-based clustering approach for database partitioning”, IEEE Transactions on Systems, Man and Cybernetics, Part C, Vol.32, Issue 3, pp.215-230 (2002)
20) Yan, X., Liu, H., Yan, J., Wu, Q.:“A Fast Evolutionary Algorithm for Traveling Salesman Problem”, Proc. of Third International Conference on Natural Computation (ICNC 2007), Vol.4, pp.85-90 (2007)
21) G. Gutin, A. P. Punnen (Eds.): “The Traveling Salesman Problem and its Variations”, Springer, NY, USA, (2007).
22) 坪井竹彦, 高橋洋二, 兵藤哲朗, "物流業務の実態を踏まえた配送ルートの設計方法の研究," 交通工学研究発表会論文報告集, vol. 21, pp. 53-56, (2001).
23) Y. Sakurai, K. Takada, N. Tsukamoto, T. Onoyama, R. Knauf, S. Tsuruta: "Inner Random Restart Genetic Algorithm to Optimize Delivery Schedule", Proc. of the 2010 IEEE International Conference on Systems, Man and Cybernetics (SMC 2010), Istanbul, Turkey, pp.263-270, (2010.10).