並列化に適した遺伝的ローカルサーチによる非線形関数最適化

木村 周平*・高橋 治**

小林 重信***・小長谷 明彦*

Non-linear Function Optimization Using a Genetic Local Search Suitable for Parallelization

Shuhei KIMURA*, Osamu TAKAHASHI**,
Shigenobu KOBAYASHI*** and Akihiko KONAGAYA*

In this paper, we propose a new genetic local search named GLSDC (a Genetic Local Search with distance independent Diversity Control) by extending the basic idea of DIDC (a genetic algorithm with Distance Independent Diversity Control) to coarse grained parallelization. GLSDC employs a local search method as a search operator. GLSDC also uses genetic operators, i.e., a crossover operator and a generation alternation model. However, in GLSDC, the crossover operator is not used as a search operator, but is used only for converging the population. GLSDC has an ability to find multiple optima simultaneously by stacking good individuals that have been found by the local search. Finding multiple optima is often required when we try to solve real world problems. The effectiveness of the proposed method is verified through numerical experiments on several benchmark problems.

Key Words: genetic local search, real-coded GA, function optimization, parallel computing, niching

1. はじめに

遺伝子として実数値ベクトルを使用する実数値遺伝的アルゴリズム (GA) や進化戦略 (ES) は、実問題における困難な関数最適化問題に対する効率的な近似解法として期待されており、さまざまな手法が提案されている^{1), 3), 17)}。実問題における関数最適化問題の持つ特徴的な性質として多峰性、変数間依存性、悪スケール性が考えられ、困難とされる実問題にはこれらの性質がしばしば現れる¹⁵⁾。なお多峰性とは複数の局所最適解が存在する性質のこと、変数間依存性とは目的関数を各変数ごとの関数の和として分離して表現することのできない性質のことである。また悪スケール性とは変数ごとのスケールが大きく異なる性質のことである。以上のような困難な性質を考慮した手法として ENDX¹⁰⁾, SPX⁶⁾, UNDX- m ¹⁴⁾, 相関突然変異¹⁾などが提案されてい

る。しかしこれらの手法は関数の探索次元が高くなると、単純なベンチマーク関数においても最適解の発見に失敗することがあった。

多峰性、変数間依存性、悪スケール性を持ち、なおかつ比較的高次元の関数最適化問題を解くための手法として、著者らは距離に依存せずに多様性を制御する GA (DIDC, a genetic algorithm with Distance Independent Diversity Control)¹¹⁾を提案している。DIDC は集団分布を維持・拡大できる GA と集団の収束を促す GA という二種類の異なる性質を持つ GA を状況に応じて使い分ける実数値 GA である。DIDC は性質の異なる二種類の GA を使い分けることで集団の多様性を制御し、これによって関数最適化問題における最適解の発見確率を高めている。また DIDC の遺伝的オペレータには、変数間依存性や悪スケール性の影響を受けにくいものが使用される。以上により DIDC は多峰性、変数間依存性、悪スケール性を持つ高次元関数最適化問題において良好な最適解発見性能を示す。ところが DIDC による関数最適化は、従来手法よりも多くの関数評価を必要とした。

計算コストの問題を克服するための方法の一つとして並列処理がある。DIDC はマスタースレーブモデル⁹⁾を利用することで、容易に並列化することができる。マスタースレーブモデルとは、選択・交叉といった遺伝的操作をマスタースレーブで行ない、他のスレーブノードでは関数の評価値のみを計算するという並列計算モデルである。しかし DIDC の

* 理化学研究所ゲノム科学総合研究センター
横浜市鶴見区末広町 1-7-22

** 理工学振興会/東京工業大学 横浜市緑区長津田町 4259

*** 東京工業大学 横浜市緑区長津田町 4259

* RIKEN Genomic Sciences Center, 1-7-22 Suehiro-cho, Tsurumi, Yokohama

** The Circle for the Promotion of Science and Engineering/Tokyo Institute of Technology, 4259 Nagatsuta-cho, Midori, Yokohama

*** Tokyo Institute of Technology, 4259 Nagatsuta-cho, Midori, Yokohama

(Received July 4, 2003)

(Revised November 10, 2003)

マスタースレーブモデルによる並列化は、PC クラスタのようなノード間の通信コストの高い並列コンピュータには適さない。これは最適化に多くの関数評価を必要とする DIDC を、関数評価のたびに通信するマスタースレーブモデルで実装すると、非常に多くの通信が必要となるためである。

そこで本論文では DIDC の基本的なアイデアを利用することにより、PC クラスタなどの通信コストの高い並列コンピュータでの実装に適した新たな最適化手法として、距離に依存せずに多様性を制御する遺伝的ローカルサーチ (GLSDC, a Genetic Local Search with distance independent Diversity Control) を提案する。以下2章ではまず DIDC について説明を行なう。DIDC の基本的なアイデアを利用することで、3章において新たな最適化アルゴリズム GLSDC を提案する。4章では GLSDC の計算コストを削減するための方法を説明する。GLSDC の性能を評価するために5章では数値実験を行なう。6章は考察、7章は結論と今後の課題である。

2. 距離に依存せずに多様性を制御する GA

本章では距離に依存せずに多様性を制御する GA(DIDC, a genetic algorithm with Distance Independent Diversity Control)¹¹⁾ のアルゴリズム、およびその探索の特徴について説明する。

2.1 DIDC による探索

GA や ES といった進化的手法を関数最適化問題に適用する場合には、集団の多様性を維持することが重要である¹³⁾。集団の多様性を積極的に維持するための方法として、Fitness Sharing⁴⁾ や DDA²⁴⁾ といった、集団分布を考慮する手法がある。これらの手法は個体間の距離を考慮することにより、集団分布を直接的に制御する。ところがこれらの手法は個体間の距離を利用しているため、悪スケール性の強い関数最適化問題では性能が悪化する場合があった。

これに対して DIDC¹¹⁾ は、集団分布を維持・拡大できる GA と集団の収束を促す GA という二種類の異なる性質を持つ GA を状況に応じて使い分けることで、距離を利用することなく集団分布を制御する。また DIDC は遺伝的オペレータとして、変数間依存性や悪スケール性の影響を受けにくいものを使用する。以上により DIDC は多峰性、変数間依存性、悪スケール性を持つ高次元関数最適化問題において良好な探索性能を示す。

DIDC では、集団分布を維持・拡大できる GA のための探索オペレータとして突然変異 NDM(Normal Distribution Mutation, 付録 B を参照)¹¹⁾ を、集団の収束を促す GA のための探索オペレータとして交叉 ENDX(Extended Normal Distribution Crossover, 付録 A を参照)¹⁰⁾ をそれぞれ使用している。また両 GA の世代交代モデルには MGG(Minimal Generation Gap model, 付録 C を参照)²¹⁾ を使用している。DIDC のアルゴリズムは以下のとおりである。

[アルゴリズム: DIDC]

Fig. 1 Framework of optimization process by DIDC. This figure shows several generations of DIDC.

- (1) 世代数を 0 に設定。初期集団として n_p 個体をランダムに生成。突然変異適用確率 $p_m = 1$ とする。
- (2) 確率 p_m でステップ 3 へ、さもなければステップ 4 へいく。
- (3) NDM/MGG に基づく世代交代を一回だけ行ない、ステップ 5 へ。
- (4) ENDX/MGG に基づく世代交代を一回だけ行ない、ステップ 5 へ。
- (5) 過去の T_s 世代の間に集団中の最良個体が更新されていなければ $p_m = 0$ とする。集団中の最良個体の評価値が僅かでも改善された場合はすぐに $p_m = 1$ とする。
- (6) 終了条件を満たしていれば終了。さもなければ世代数を 1 増加させてステップ 2 へ戻る。

パラメータ $T_s = 3n_p$ が推奨されている。ただし n_p は集団サイズである。

2.2 探索フェーズの切り替えによる多様性の制御

DIDC による最適化過程は主に、NDM/MGG に基づく探索と ENDX/MGG に基づく探索という二つの探索フェーズから構成されていると考えることができる (Fig. 1)。

NDM/MGG は主に各個体の近傍を探索する GA であり、局所探索の性質を持つ。そのため NDM/MGG を多峰性関数に適用すると、集団中の各個体は異なる局所解の谷に捉われる。以上のために NDM/MGG は多峰性関数において集団分布を収束させることができず、効率的な探索を行なうことができないという問題を持つ。DIDC は逆にこの性質を利用することで、多峰性関数において集団の多様性を維持する。なお NDM/MGG は単に局所探索の性質を持っているだけであり、常に集団の多様性を維持する性質を持っているわけではないという点に注意が必要である。実際、単峰性関数において NDM/MGG は集団を収束させる。しかし単峰性関数のような単純な関数において、集団の多様性を維持する必要は必ずしもないと考えられる。本論文では以上の

ように局所的な探索を行なうフェーズを、局所探索フェーズ (local search phase) と呼ぶことにする。他方 ENDX/MGG は集団を強制的に収束させることができる。しかし集団の過度な収束のために、望ましい領域への収束に失敗する場合がある。このように集団の収束を促すフェーズを、本論文では収束フェーズ (converging phase) と呼ぶ。DIDC は以上のような性質の異なる二種類の GA に基づく探索フェーズを状況に応じて切り替えることで、集団分布を適切に狭めながら探索を行なう。

DIDC の探索においては、上述のように適切な性質を持つ探索フェーズを状況に応じて適切に切り替えることこそが重要である。したがってそれぞれの探索フェーズで使用する探索器の選択は二次的なことからであり、必要な性質さえ満たしていればどのような探索器を使用しても良いと考えられる。以上の考えから次章では、二つのフェーズを再構成することにより、新たな最適化アルゴリズム GLSDC を提案する。

3. GLSDC の提案

DIDC はマスタースレーブモデルを用いることで、容易に並列化できる。しかし 1 章で述べたように DIDC は PC クラスタなどの通信コストの高い並列コンピュータへの実装には適さない。そこで本章では DIDC の基本的なアイデアを利用することにより、並列化に適した新たな探索手法として距離に依存せずに多様性を制御する遺伝的ローカルサーチ (GLSDC, a Genetic Local Search with distance independent Diversity Control)¹²⁾ を提案する。

3.1 設計方針

2.2 節で述べたように DIDC の探索においては二種類の探索フェーズを状況に応じて適切に切り替えることだけが重要であるとの立場から、それぞれの探索フェーズをその機能を変えることなく並列化を考慮しながら設計し直すことを考える。並列化にはマスタースレーブモデル⁹⁾を採用するものとし、ノード間の通信頻度を低減させることで並列コンピュータにおける性能向上を目指す。

局所探索フェーズとして、NDM/MGG を使用する代わりに集団中のすべての個体に既存のローカルサーチ手法を適用することを考える。前述したように NDM/MGG は局所探索の性質を持つため、これを既存のローカルサーチ手法に置き換えた場合にも局所探索フェーズの機能は変わらないと考えられる。このフェーズは、ローカルサーチの計算を各スレーブノードにおいて実行することで容易に並列化できる。またこのようにスレーブノードにおける計算を一個体の評価値計算からローカルサーチの計算にすることで、一回の通信当りのスレーブノードにおける計算量が大きくなり、結果として通信頻度の低減も期待できる。

収束フェーズには DIDC と同様に、ENDX/MGG を使用する。ただしここでは ENDX/MGG の持つ集団を収束させる能力だけに注目し、解の改善は期待しないものとする。す

Fig. 2 Generation alternation by GLSDC

なわち交叉 ENDX によって新しく生成した子個体の評価値は計算せず、代わりに親集団中のどの個体よりも悪い評価値を持つものとして扱う。子個体の評価値を計算しない場合でも ENDX/MGG は集団分布を収束させる性質を持つため、GLSDC における収束フェーズの機能も DIDC のそれと変わらない。なお子個体の評価値を計算しない ENDX/MGG が集団を収束させる性質を持つことに関しては、6.2 節において実験的に確認する。このフェーズをマスタースレーブモデルによって効率的に並列化することは難しい。しかし個体の評価値を計算しないため、並列化しなくても高速に実行できると考えられる。

本論文では、以上の設計方針に従って GLSDC を提案する。なおこの手法はローカルサーチ手法と遺伝的オペレータを併用するため、遺伝的ローカルサーチと呼ぶ。進化的手法の並列化に関しては多くの研究がなされている^{2), 5), 7), 9), 16)}。GLSDC に似た手法として ASPARAGOS²²⁾ が提案されている。しかし特に ASPARAGOS の対象が組み合わせ最適化問題であるという点、そして GLSDC が集団を収束させるためだけに遺伝的オペレータを使用しているという点で、両者は異なる。

3.2 GLSDC のアルゴリズム

GLSDC のアルゴリズムを以下に示す (Fig. 2)。ただし以下のアルゴリズムにおいては、離散的な世代交代を行なうモデルが採用されている点に注意が必要である。これに対して DIDC は連続的な世代交代を行なうモデルを採用している。したがって DIDC の一世代と GLSDC の一世代では、世代交代する個体の数が大きく異なる。

[アルゴリズム: GLSDC]

(1) 世代数を 0 に設定。初期集団として n_p 個体をランダムに生成。ENDX/MGG 適用回数 $N_{iter} = N_0$ とする。

(2) [局所探索フェーズ]

集団中のすべての個体にローカルサーチを適用する。

(3) [N_{iter} の適応]

もし局所探索フェーズでの探索で集団中の最良個体が、前の世代と比較して改善されなかった場合 $N_{iter} \leftarrow N_{iter} + N_0$ とする。逆に集団中の最良個体が僅かでも改善された場合には、 $N_{iter} = N_0$ にもどす。

(4) [収束フェーズ]

ENDX/MGGに基づく世代交代操作を N_{iter} 回繰り返す。ただしこのとき生成子個体の評価値は計算しない。

(5) 終了条件を満たしていれば終了。さもなければ世代数を1増加させてステップ2へ戻る。

GLSDCのパラメータ N_0 は予備の実験に基づいて $N_0 = 2n_p$ とする。ただし n_p は集団サイズである。

本章の残りで GLSDC のアルゴリズムにおける局所探索フェーズ、 N_{iter} の適応、収束フェーズの3つのステップに関する説明を行なう。

3.2.1 ステップ：局所探索フェーズ

本ステップでは集団中のすべての個体に既存のローカルサーチ手法を適用する。どのような関数最適化手法も本ステップのローカルサーチ手法として利用可能である。本論文ではローカルサーチ手法として、修正 Powell 法 (modified Powell's method)¹⁸⁾ を使用する。修正 Powell 法は方向集合法と呼ばれる探索法の一つであり、GA と同様に解の探索のために勾配情報を利用しないという特徴を持つ。

本ステップは GLSDC において最も多くの計算を必要とする。しかし 3.1 節で説明したように、マスタースレーブモデルを利用することにより並列処理が可能である。本ステップに関しては、4 章においてさらに詳しく説明する。

3.2.2 ステップ： N_{iter} の適応

GLSDC は収束フェーズにおいて ENDX/MGG を適用することで、集団分布を僅かに狭める。このとき、集団分布の縮小率は ENDX/MGG 適用回数 N_{iter} に依存する。もし局所探索フェーズにおける探索で集団中の最良個体が改善されなかった場合、GLSDC は集団の収束が不十分と判断して N_{iter} を増加させる。

3.2.3 ステップ：収束フェーズ

集団を収束させるために、本ステップでは ENDX/MGG に基づく世代交代操作を N_{iter} 回繰り返す。ただしこのとき 3.1 節で述べたように、ENDX によって新たに生成された子個体の評価値は計算しない。代わりにこれらの個体は親集団中のどの個体よりも悪い評価値を等しく持つものとして扱う。このステップをマスタースレーブモデルによって効率的に並列化することは難しい。しかし個体の評価値を計算しないため、並列化しなくても高速に実行できると考えられる。

MGG による毎回の世代交代操作では、家族中から選択された2個体とその親個体と入れ替わる (付録 C を参照)。なお家族とは交叉における親個体とその子個体のことである。親と入れ替わる2個体のうちの1個体は家族中の最良個体であり、もう1個体は評価値とは無関係に家族中からランダムに選択される。したがって MGG による世代交代操作に

おいては、最良個体を除く家族中のどの個体も評価値とは無関係に等しく扱われる。他方、GLSDC において親集団にはすでにローカルサーチ手法が適用されている。したがって親集団中の個体は比較的良好な評価値を持つはずである。交叉は確率的探索オペレータであるため、このような良い評価値を持つ個体を親として子個体を生成した場合には、生成子個体が親個体よりも良い評価値を持つ可能性は低いと考えられる。以上のことから GLSDC において、ENDX/MGG に基づく世代交代操作の際の生成子個体は、評価値とは無関係に扱われる可能性が高いことがわかる。そこで本研究では計算コスト削減の観点から、ENDX によって新たに生成された子個体の評価値を計算せず、親集団中のどの個体よりも悪い評価値を等しく持つものとして扱うことにした。

4. 計算コストを考慮した局所探索フェーズの設計

GLSDC は局所探索フェーズにおいて、ローカルサーチを繰り返し適用する。したがって効率的な探索のためには、ローカルサーチの計算コスト削減が重要である。以上の観点から本章では、局所探索フェーズにおけるローカルサーチの計算コスト削減を試みる。

4.1 局所探索フェーズの計算コスト削減

以下の三種類の接近法により、局所探索フェーズにおけるローカルサーチの計算コストを削減する。

4.1.1 探索範囲の適応

GLSDC は局所探索フェーズと収束フェーズを交互に繰り返すことで、徐々に集団分布を狭めながら探索を行なう。これはすなわち、GLSDC が解の改善の見込みのない領域の探索を諦め、徐々に有望な領域の探索に計算資源を集中させることを意味する。したがって GLSDC において集団が十分に収束したあとにローカルサーチによって広い領域を探索することは、見込みがないとして探索を打ち切った領域を再び探索することであり、無駄が大きいと考えられる。以上の理由により GLSDC において効率的な探索を行なうためには、状況に応じて探索範囲を変えられるローカルサーチ手法を使用することが望ましい。

本論文で使用する修正 Powell 法は初期方向セット (initial direction set) と呼ばれる n 本の n 次元ベクトルからなる制御パラメータを利用することで、探索範囲を変化させることが可能である。ただし n は探索空間の次元数である。本論文では n 本のベクトルを探索空間の各軸方向にとる。また各ベクトルの大きさは、その方向に関する集団の分布範囲に基づいて設定する。以上のような初期方向セットの設定を各世代ごとに行なうことで、GLSDC は修正 Powell 法の探索範囲を集団分布に応じて変化させることができる。

4.1.2 探索打ち切り条件の切り替え

一般にローカルサーチにおいて、より多くの関数評価を許せば、より良い解の発見が期待できる。しかし見込みのない領域の探索に多くの計算リソースを注ぎ込むことは無駄が大きい。そこで見込みのないと思われる領域の探索において

は、少ない関数評価回数でローカルサーチを打ち切ることを考える。

上述のような探索を実現するために、ローカルサーチに適用する探索打ち切り条件として精度基準と評価回数基準の二種類を考える。精度基準はローカルサーチに対して、何らかの局所解を高い精度で発見するのに十分な探索を許すものである。また評価回数基準は、ローカルサーチにおいて関数評価回数があらかじめ決められた上限に達したら探索を打ち切るものである。本論文ではローカルサーチの適用される個体群のなかで最良評価値を持つ個体に対しては精度基準を、それ以外の個体に対しては評価回数基準を探索打ち切り条件としてローカルサーチを適用することで計算コストを削減する。

4.1.3 不要なローカルサーチの省略

前項で述べた精度基準は、ローカルサーチに対して何らかの局所解を発見するのに十分な探索を許す。したがって精度基準に基づくローカルサーチを適用された個体は、すでに何らかの局所解に到達しているものと考えられる。このような個体に対して再びローカルサーチを適用することは無駄である。そこでこのような個体に対してはローカルサーチの適用を省略することで、計算コストを削減する。

4.2 局所探索フェーズのアルゴリズム

前節で説明した接近法に基づき、計算コストの削減を考慮した局所探索フェーズの手続きを以下に示す。

[アルゴリズム: 局所探索フェーズ]

- (1) 集団分布に応じてローカルサーチ手法の探索範囲を更新する。
- (2) 集団を候補個体群と充分個体群に分類する。充分個体とは過去に精度基準に基づくローカルサーチを適用され、すでに何らかの局所解に到達していると考えられる個体のことである。これらの充分個体群に対してはローカルサーチの適用を省略する。また候補個体とは、充分個体ではない個体のことである。
- (3) 候補個体群のなかで最良の評価値を持つ個体に対して、精度基準に基づくローカルサーチを適用する。
- (4) 残りの候補個体に対しては、評価基準に基づくローカルサーチを適用する。

本論文では評価回数基準におけるローカルサーチの関数評価回数の上限を $100n$ とした。ただし n は探索空間の次元数である。

5. 数値実験

本章では数値実験を通して GLSDC の性能を評価する。

5.1 対象問題

Table 1 に示す関数の最小化を目的として実験を行なう。ただしすべての関数において GA の初期個体を Table 1 に示される探索領域内に限定して生成し、探索中は探索領域を明示的には考慮しないものとする。実験は 10 次元 ($n = 10$)、30 次元 ($n = 30$) および 100 次元 ($n = 100$) の関数に対し

て行なう。

Rosenbrock 関数は変数間依存性の強い単峰性関数であり、ill-scaled Rosenbrock 関数は変数ごとにスケールを変化させた Rosenbrock 関数である。両者の違いは変数ごとのスケールだけであり、そのほかの性質は全く同じである。

Griewangk 関数は多峰性関数であるが、次元数の増加と共に Sphere 関数に近くなり最適化が容易となるという性質を持つ。Ackley 関数、Rastrigin 関数および shifted Rastrigin 関数は多峰性の強い関数である。なお Rastrigin 関数では初期集団の初期化領域を $-100 \leq x_i \leq 100 (i = 1, \dots, n)$ とする。これは通常の設定よりも広く、難しい問題となっている。また shifted Rastrigin 関数は、Rastrigin 関数における初期集団の初期化領域を 10% ずらした関数である。これによりサンプリング・バイアスによって探索領域の中央に位置する最適解の発見確率が高くなる影響²³⁾を緩和できると考えられる。

5.2 実験 1: 探索性能の比較

本節では Table 1 に示されるベンチマーク問題を用いて、提案手法と代表的な従来手法の探索性能を比較する。

5.2.1 実験設定

本論文では GLSDC の集団サイズ $n_p = 3n$ とする。ただし n は関数の次元数である。一般に遺伝的ローカルサーチは、GA よりも少ない集団サイズで GA と同等の性能を示す。また収束フェーズにおいて、交叉 ENDX によって一度に生成される子個体数 $n_c = 10$ とする。なお ENDX は多数の親 (m 個体) を使用する交叉オペレータであり、子個体の生成には $O(mn)$ の時間計算量を必要とする。ENDX の親個体数として $m = n + 2$ が推奨されているが、本論文では計算量削減の観点からすべての実験で $m = 12$ とする。以上の設定は予備実験に基づいて決定した。

比較のための GA には DIDC¹¹⁾、SPX⁶⁾/MGG、UNDX- m ¹⁴⁾ + EDX¹⁹⁾/MGG、島モデル GA(IGA, Island GA)⁸⁾ を使用する。各 GA の制御パラメータは、各文献を参考に以下のように決定する。DIDC では集団サイズ $n_p = 15n$ 、生成子個体数 $n_c = 50$ とする。また上述したように計算量削減の観点から ENDX の親個体数を $m = 12$ とする。同じ理由により NDM の親個体数も $m = 12$ とする。なお親個体数を減らしても、探索性能にはほとんど変化が見られなかった。SPX/MGG では集団サイズ $n_p = 30n$ 、生成子個体数 $n_c = 200$ とする。UNDX- m + EDX/MGG は DIDC と同様に二種類の性質の異なる探索オペレータを使用した GA であり、稜構造を持つ高次元関数最適化において良い性能を示すことが報告されている¹⁹⁾。UNDX- m + EDX/MGG の集団サイズ $n_p = 15n$ 、UNDX- m による生成子個体数 $n_{cu} = 200$ 、EDX による生成子個体数 $n_{ce} = 100$ 、EDX 適用確率 0.5 とする。また UNDX- m のパラメータ $m = \min\{20, n/2\}$ とする。

島モデル GA(IGA) は並列処理に適した GA である。IGA は集団を島と呼ばれるいくつかのサブ集団に分割し一定世

Table 1 Benchmark functions

Objective function		Search region	Optimum
Rosenbrock	$f_{ro}(\mathbf{x}) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$-2.048 \leq x_i \leq 2.048$	$f_{ro}(1, \dots, 1) = 0$
ill-scaled Rosenbrock	$f_{iro}(\mathbf{x}) = \sum_{i=1}^{n-1} \{100[(i+1)x_{i+1} - (ix_i)^2]^2 + (ix_i - 1)^2\}$	$-\frac{2.048}{i} \leq x_i \leq \frac{2.048}{i}$	$f_{iro}(\frac{1}{1}, \dots, \frac{1}{n}) = 0$
Griewangk	$f_{gr}(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$-512 \leq x_i \leq 512$	$f_{gr}(0, \dots, 0) = 0$
Ackley	$f_{ac}(\mathbf{x}) = 20 + e - 20 \exp\left(-0.2 \sqrt{\frac{\sum_{i=1}^n x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=1}^n \cos 2\pi x_i}{n}\right)$	$-20 \leq x_i \leq 30$	$f_{ac}(0, \dots, 0) = 0$
Rastrigin	$f_{ra}(\mathbf{x}) = 10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)]$	$-100 \leq x_i \leq 100$	$f_{ra}(0, \dots, 0) = 0$
shifted Rastrigin	$f_{sra}(\mathbf{x}) = 10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)]$	$-80 \leq x_i \leq 120$	$f_{sra}(0, \dots, 0) = 0$

Table 2 Summary of results

Objective function	Dimension n	GLSDC	DIDC	SPX/MGG	UNDX- m +EDX	IGA1	IGA8	IGA16
		SUC (EVAL)	SUC (EVAL)	SUC (EVAL)	SUC (EVAL)	SUC (EVAL)	SUC (EVAL)	SUC (EVAL)
Rosenbrock	10	10/10 (8.80×10^4)	10/10 (1.05×10^6)	10/10 (1.77×10^6)	10/10 (9.35×10^5)	6/10 (1.97×10^6)	5/10 (1.93×10^6)	1/10 (1.96×10^6)
	30	10/10 (7.46×10^5)	10/10 (1.67×10^7)	0/10 —	10/10 (9.74×10^6)	0/10 —	0/10 —	0/10 —
	100	10/10 (1.17×10^7)	0/10 —	0/10 —	0/10 —	0/10 —	0/10 —	0/10 —
ill-scaled Rosenbrock	10	10/10 (6.70×10^4)	10/10 (1.00×10^6)	10/10 (1.76×10^6)	10/10 (1.10×10^6)	0/10 —	0/10 —	0/10 —
	30	10/10 (6.19×10^5)	8/10 (1.60×10^7)	0/10 —	10/10 (1.46×10^7)	0/10 —	0/10 —	0/10 —
	100	10/10 (1.18×10^7)	0/10 —	0/10 —	0/10 —	0/10 —	0/10 —	0/10 —
Griewangk	10	10/10 (4.20×10^4)	10/10 (1.05×10^6)	10/10 (1.40×10^6)	10/10 (6.71×10^5)	10/10 (6.32×10^5)	10/10 (6.35×10^5)	10/10 (6.81×10^5)
	30	10/10 (2.93×10^5)	10/10 (6.89×10^6)	10/10 (5.18×10^6)	10/10 (2.66×10^6)	10/10 (2.71×10^6)	10/10 (2.70×10^6)	10/10 (2.68×10^6)
	100	10/10 (2.83×10^6)	10/10 (6.04×10^7)	10/10 (2.75×10^7)	10/10 (1.90×10^7)	10/10 (2.70×10^7)	10/10 (2.73×10^7)	10/10 (2.71×10^7)
Ackley	10	10/10 (1.80×10^4)	10/10 (1.30×10^6)	10/10 (1.63×10^6)	10/10 (6.09×10^5)	10/10 (5.84×10^5)	10/10 (5.82×10^5)	10/10 (5.90×10^5)
	30	10/10 (1.57×10^5)	10/10 (1.14×10^7)	10/10 (7.45×10^6)	10/10 (3.61×10^6)	10/10 (3.21×10^6)	10/10 (3.21×10^6)	10/10 (3.21×10^6)
	100	10/10 (1.58×10^6)	10/10 (8.98×10^7)	10/10 (3.89×10^7)	10/10 (2.40×10^7)	10/10 (2.77×10^7)	10/10 (2.79×10^7)	10/10 (2.79×10^7)
Rastrigin	10	10/10 (5.40×10^4)	10/10 (1.29×10^6)	10/10 (1.65×10^6)	9/10 (1.03×10^6)	10/10 (8.90×10^5)	10/10 (8.87×10^5)	10/10 (9.19×10^5)
	30	10/10 (1.39×10^6)	9/10 (9.72×10^6)	10/10 (8.23×10^6)	0/10 —	10/10 (4.37×10^6)	9/10 (4.28×10^6)	9/10 (4.29×10^6)
	100	10/10 (3.47×10^7)	10/10 (7.70×10^7)	1/10 (4.63×10^7)	0/10 —	0/10 —	0/10 —	0/10 —
shifted Rastrigin	10	10/10 (1.11×10^5)	10/10 (1.26×10^6)	10/10 (1.67×10^6)	9/10 (1.08×10^6)	10/10 (9.44×10^5)	10/10 (1.07×10^6)	10/10 (9.92×10^5)
	30	10/10 (2.06×10^6)	10/10 (9.52×10^6)	9/10 (8.25×10^6)	0/10 —	1/10 (9.36×10^6)	0/10 —	1/10 (9.18×10^6)
	100	10/10 (5.09×10^7)	10/10 (7.56×10^7)	0/10 —	0/10 —	0/10 —	0/10 —	0/10 —

代ごとに島間で個体の交換を行ないながら、それぞれの島で独立して GA を実行する。IGA は並列コンピュータにおいて、それぞれの島をそれぞれの計算ノードに割り当てることのできるため、効率的な並列処理が可能である。本論文では島数を 1(IGA1), 8(IGA8), 16(IGA16) とした場合の実験を行なう。なお島数が 1 の IGA は、通常の単一集団の GA と同じである。集団サイズ $n_p = 32n$ 、一回の交叉あたりの生

成子個体数 $n_c = 50$ 、移住率 0.5、移住間隔 5 とし、各島では UNDX¹⁷⁾/MGG に基づく世代交代を行なうものとする。

各 GA に関するそのほかの制御パラメータは、すべて各文献の推奨パラメータを使用する。多くの文献では単峰性関数と多峰性関数で、集団サイズを変えて実験が行なわれている。しかし一般に実問題において適応度景観は不明であることから、本論文ではどちらも同じ集団サイズとした。

探索の終了は (1) 評価値が 1.0×10^{-6} に達した, (2) 関数評価回数が $2 \times 10^4 \times n^2$ に達した, (3) 集団がすべての座標軸に関して 10^{-6} 以下の領域に収束した, (4) 2×10^7 回の関数評価にわたって評価値が改善しなかった, のいずれかの条件が満たされた場合とする. また評価値が 1.0×10^{-6} に達した場合にのみ, 最適解を発見したと見なす.

5.2.2 結果

それぞれの手法を各ベンチマーク関数に適用したときの最適解発見率 (SUC) と, 最適解発見までに要した関数評価回数の平均 (EVAL) を Table 2 に示す.

GLSDC は洗練された既存のローカルサーチ手法を利用するため, Rosenbrock 関数のように変数間依存性の強い関数であっても単峰性であれば, 非常に少ない関数評価回数で最適解を発見することができる. これに対して SPX/MGG は次元の高い Rosenbrock 関数では初期収束を起こし, 最適解の発見に失敗した. また DIDC, UNDX- m + EDX/MGG, IGA は最適化に多くの関数評価を必要とするため, 高次元の Rosenbrock 関数では探索打ち切り世代までに最適解を発見することができなかった. GLSDC, DIDC, SPX/MGG は悪スケール性の影響を受けにくいいため, ill-scaled Rosenbrock 関数においても Rosenbrock 関数と同様の探索性能を示した. 他方 UNDX- m + EDX/MGG と IGA は悪スケール性の影響を受けるため, ill-scaled Rosenbrock 関数では探索性能が悪化した. 多峰性関数においては GLSDC と DIDC だけがすべての関数で最適解を発見することができた. さらに GLSDC は多峰性関数においても, 他手法よりも少ない関数評価回数での最適解の発見に成功した. GLSDC は既存のローカルサーチ手法として修正 Powell 法を利用しているため, 特にベンチマーク関数のような連続な関数においては多峰性関数であっても, 他手法よりも少ない関数評価回数で効率的に最適化できる.

以上より GLSDC は変数間依存性, 悪スケール性, 多峰性を持つ連続な関数最適化において良好な探索性能を示すことがわかる. またその探索性能は, DIDC と同等以上であった.

5.3 実験 2: 並列処理による計算時間の比較

つぎに本節では提案手法の性能を, 並列コンピュータにおける計算時間の観点から評価する.

5.3.1 実験設定

GLSDC, DIDC および IGA を, 1 CPU のコンピュータ (PentiumIII 933MHz) および PC クラスタ (PentiumIII 933MHz $\times$ 8 CPUs, PentiumIII 933MHz $\times$ 16 CPUs) で実行したときの計算時間を比較する. なお IGA では集団を CPU 数に等しい数の島に分割するものとする. IGA の並列化は, それぞれの島をそれぞれの CPU に割り当てることで実現する. また GLSDC と DIDC の並列化には, マスタスレーブモデルを利用する. 目的関数には 100 次元 Rastrigin 関数 (Table 1) を用いるものとし, 関数評価回数が 1.0×10^7 回に達するまでの時間を比較する. ほかの実験

Table 3 Average computational time required for reaching 1.0×10^7 function evaluations on a 100 dimensional Rastrigin function by running GLSDC, DIDC and IGA on a single CPU computer, 8 CPUs PC cluster and 16 CPUs PC cluster.

	1 CPU	8 CPUs	16 CPUs
GLSDC	241 (sec.)	57 (sec.)	36 (sec.)
DIDC	1062 (sec.)	1532 (sec.)	1638 (sec.)
IGA	1326 (sec.)	456 (sec.)	179 (sec.)

設定は前節と同じとする.

5.3.2 結果

それぞれのコンピュータにおいて GLSDC, DIDC および IGA を 100 次元 Rastrigin 関数に適用したときに, 関数評価回数が 1.0×10^7 回に達するまでに要した計算時間の平均を Table 3 に示す. DIDC では並列化したほうが, かえって同じ回数の関数評価に多くの計算時間を必要とした. 特にベンチマーク問題のように目的関数の計算コストが低い場合, DIDC を PC クラスタのような通信コストの高いコンピュータで並列化しても効果がないことがわかる. これに対して GLSDC と IGA は並列化により, 計算時間を短縮させることができた. なお IGA はすべての実験において, GLSDC よりも多くの計算時間を必要とした. この理由は, 遺伝的オペレータの計算コストが既存のローカルサーチ手法と比較して高いためであると考えられる. GA はその探索において遺伝的オペレータを繰り返し使用する. したがってベンチマーク問題のように目的関数の計算コストが低い場合には, 遺伝的オペレータの計算コストの影響は大きい. また 1CPU のコンピュータにおいて DIDC の要した計算時間が GLSDC よりも多い理由も, 同様に遺伝的オペレータの計算コストの影響であると考えられる.

DIDC をマスタスレーブモデルで並列化する場合, 関数評価回数と同じオーダの通信が必要である. 他方 GLSDC では, 集団サイズ $\times$ 世代数のオーダの通信が必要となる. ただし一回当たりの通信量はどちらも高々 $O(n)$ である. 本実験では関数評価回数は 1.0×10^7 であり, このとき GLSDC の必要とした世代数は平均で 10.0 世代であった. GLSDC の集団サイズは $n_p = 3n$ であるから, DIDC と比較すると非常に少ない通信回数であることがわかる. このため GLSDC では PC クラスタのような通信コストの高い並列コンピュータにおいても, 効率的に並列処理が行なわれる.

本実験では, 同じ回数の関数評価時間を比較しているという点に注意が必要である. GLSDC は前節の実験で示したように連続な関数では他手法より少ない関数評価回数で最適解を発見できるため, 最適解発見までの計算時間の差はさらに大きい. なお IGA は 100 次元 Rastrigin 関数において最適解を発見することができなかった.

6. 考察

6.1 使用するローカルサーチ手法について

3.2.1 項で説明したように, どのような関数最適化手法も

GLSDC のローカルサーチ手法として使用できる。しかし GLSDC が良好な性能を発揮するためには、適用する関数に応じて適切なローカルサーチ手法を選択する必要がある。

本論文で使用した修正 Powell 法は解の探索のために勾配情報を利用しないという特徴を持つ。しかし修正 Powell 法はその探索において関数の連続性を暗黙の仮定としているため、至る所で不連続な関数に対する使用は困難であると考えられる。したがってローカルサーチ手法として修正 Powell 法を使用している GLSDC を、至る所で不連続な関数に適用することは難しい。しかしながら実問題においては、ほとんどの領域で連続であるような関数最適化問題も多いものと考え、本研究では GLSDC のローカルサーチ手法として修正 Powell 法を使用した。

5.2 節の実験において提案手法が他手法と比較して特に少ない関数評価回数で最適化を行なうことのできた理由は、連続なベンチマーク関数に対して関数の連続性を利用する修正 Powell 法を用いた手法を適用したためである。至る所で不連続な関数最適化問題に GLSDC を適用するためには、ローカルサーチ手法として進化的手法や SA(Simulated Annealing) といった不連続な関数に対しても適用可能な関数最適化手法を使用する必要がある。この場合には関数の連続性を利用しないため、本論文の実験のように GLSDC が他手法よりも特に少ない関数評価回数で効率的に最適化を行なうことは期待できない。しかしながらそのような場合であっても、GLSDC の最適解発見率と並列化に関する性能は影響を受けないことが予想される。他方、目的関数が微分可能であれば、GLSDC のローカルサーチ手法として共役勾配法¹⁸⁾ のような勾配情報を利用する関数最適化手法が使用可能である。勾配情報を利用する探索手法を使用することができれば、修正 Powell 法を使用する場合よりも少ない関数評価回数での探索が期待できる。

6.2 ENDX/MGG による集団の収束について

GLSDC は収束フェーズにおいて、集団を収束させるためだけに ENDX/MGG を利用している。この際、交叉 ENDX によって新たに生成される子個体の評価値は計算されず、代わりにこれらの子個体は親集団中のどの個体よりも悪い評価値を等しく持つものとして扱われる。本節では、このように子個体の評価値を計算しない ENDX/MGG が、集団を収束させる性質を持っていることを実験的に確認する。なおここでは集団の収束を評価する指標として集団の重心と各個体間との平均距離を利用するものとし、この指標を集団の多様性 (population diversity) と呼ぶ。

Fig. 3 は 30 次元 Sphere 関数

$$f_{sp}(\mathbf{x}) = \sum_{i=1}^n x_i^2, \quad -5.12 \leq x_i \leq 5.12, \quad n = 30, \quad (1)$$

に対して生成子個体の評価値を計算しない ENDX/MGG だけを適用したときの、集団の多様性の変化の 10 試行平均である。ただし集団サイズ $n_p = 90$ 、ENDX による生成子個

Fig. 3 The population diversity plotted against the number of applying ENDX/MGG operations on 30 dimensional Sphere function

Table 4 Average number of semi-optima found by GLSDC on Rastrigin function

Dimension n	Num. of semi-optima found by GLSDC
10	3.5
30	8.8
100	9.5

体数 $n_c = 10$ とした。Fig.3 より、ENDX/MGG 適用回数の増加とともに集団が収束していることがわかる。ただしこの実験では、集団が必ずしも最適解に収束しているわけではないという点に注意が必要である。

この実験では初期集団以外の個体の評価値は一切計算されていない。したがって Sphere 関数に限らずどのような関数においても、同様の結果が得られると考えられる。以上のことから子個体の評価値を計算しない ENDX/MGG は、どのような関数においても集団を収束させる性質を持つということがわかる。

6.3 複数局所解の発見について

多峰性関数の最適化においては、大域的な最適解のほかには複数の局所解を発見することがしばしば望まれる。このような要請に対して、ニッチング (niching) と呼ばれる多くの手法が提案されている^{4), 20), 25)}。

GLSDC はローカルサーチの適用によって探索中に得られた良い評価値の解を集団とは別に保存しておくことにより、探索終了時に複数の局所解を提示することが可能である。ただしこのように解を保存するためには、ほかのニッチング手法と同様に個体間の距離を利用して個体を区別する必要がある。個体間の距離を利用するため、この操作は関数の悪スケール性の影響を受けるという点に注意が必要である。しかしながらこの操作では良好な解を集団とは別に保存する。したがって GLSDC の最適解に対する探索性能は、この操作の影響を一切受けない。

例として Rastrigin 関数において、GLSDC が 1 試行で発見した準最適解 (評価値 0.995) の平均個数を Table 4 に示す。なお n 次元 Rastrigin 関数は $2n$ 個の準最適解を持つ。本実験では距離 0.6 以上の個体を別の局所解に存在するも

のとして区別した。実験条件は5章と同じである。GLSDCはこのように1試行で複数の局所解を発見できるため、1試行で1種類の解しか発見できない一般的なGAよりも無駄の少ない探索を行なっているといえる。

7. おわりに

本論文ではDIDCの基本的なアイデアを利用することにより、並列化に適した新たな最適化アルゴリズムとして距離に依存せずに多様性を制御する遺伝的ローカルサーチ (GLSDC, a Genetic Local Search with distance independent Diversity Control) を提案した。数値実験を通してGLSDCがベンチマーク問題においてDIDCと同等以上の探索性能を持つこと、そしてGLSDCが並列化に適した手法であることを示した。また考察においてGLSDCが同時に複数の局所解を発見する能力を持つことを示した。ただし本論文においてGLSDCのローカルサーチ手法として使用されている修正Powell法が、関数の連続性を仮定しているという点に注意が必要である。したがってローカルサーチ手法として修正Powell法を使用しているGLSDCを、至る所で不連続な関数に適用することは困難である。至る所で不連続な関数最適化問題にGLSDCを適用するためには、ローカルサーチ手法として不連続な関数に対しても適用可能な関数最適化手法を使用する必要がある。

GLSDCではマスタースレーブモデルを利用して、局所探索フェーズだけが並列化される。収束フェーズの計算コストは局所探索フェーズと比較すると少ないため、収束フェーズは並列化しなくても十分に高速に実行可能である。しかしさらなる効率化のためには、収束フェーズの並列化も必要であると考えられる。島モデルGAのようなより効率的な並列化手法を参考にして、GLSDCの並列化効率を上げることが今後の課題である。

参考文献

- 1) T.Bäck, U.Hammel and H.-P.Schwefel: Evolutionary Computation: Comments on History and Current State, IEEE Trans. on Evolutionary Computation, 1-1, 3/17 (1997)
- 2) J.Digalakis and K.Margaritis: A Parallel Memetic Algorithm for Solving Optimization Problems, Proc. of 4th Metaheuristics Int. Conf., 121/125 (2001)
- 3) L.J.Eshelman and J.D.Schaffer: Real-Coded Genetic Algorithms and Interval-Schemata, Foundations of Genetic Algorithms 2, 187/202 (1993)
- 4) D.E.Goldberg and J.Richardson: Genetic Algorithms with Sharing for Multimodal Function Optimization, Proc. of 2nd Int. Conf. on Genetic Algorithms, 41/49 (1989)
- 5) V.S.Gordon and D.Whitley: Serial and Parallel Genetic Algorithms as Function Optimizers, Proc. of 5th ICGA, 177/183 (1993)
- 6) 樋口, 筒井, 山村: 実数値GAにおけるシンプレックス交叉の提案, 人工知能学会論文誌, 16-1, 147/155 (2001)
- 7) T.Hiroyasu, M.Miki and M.Ogura: Parallel Simulated Annealing using Genetic Crossover, Proc. of IASTED Int. Conf. Parallel and Distributed Computing and Systems, 139/144 (2000)
- 8) 廣安, 三木, 福永, 吉田: 実数値遺伝的アルゴリズムの分散効果の検討, 情報処理学会情報処理学会第63回全国大会講演予稿集 1, 111/112 (2001)
- 9) 廣安, 三木, 佐野, 谷村, 濱崎: 2個体分散遺伝的アルゴリズム, 計測自動制御学会論文集, 38-11, 990/995 (2002)
- 10) 木村, 小野, 喜多, 小林: 交叉の設計指針に基づくUNDXの拡張: ENDXの提案と評価, 計測自動制御学会論文集, 36-12, 1162/1171 (2000)
- 11) 木村, 小長谷: 距離に依存せずに多様性を制御するGAによる高次元関数最適化, 人工知能学会論文誌, 18-4, 193/202 (2003)
- 12) S.Kimura and A.Konagaya: High Dimensional Function Optimization using a new Genetic Local Search suitable for Parallel Computers, Proc. of 2003 Int. Conf. on Systems, Man, and Cybernetics, 335/342 (2003)
- 13) H.Kita and M.Yamamura: A Function Specialization Hypothesis for Designing Genetic Algorithms, Proc. of SMC'99, 579/584 (1999)
- 14) 喜多, 小野, 小林: 実数値GAのための正規分布交叉の多数の親を用いた拡張法の提案, 計測自動制御学会論文集, 36-10, 875/883 (2000)
- 15) 小林, 木村, 小野: 生物的適応システム～進化・学習のアルゴリズムと創発システム論～, 計測と制御, 40-10, 752/757 (2001)
- 16) H.Mühlenbein: Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization, Proc. of 3rd ICGA, 416/421 (1989)
- 17) I.Ono and S.Kobayashi: A Real-coded Genetic Algorithm for Function Optimization Using Unimodal Normal Distribution Crossover, Proc of 7th ICGA, 246/253 (1997)
- 18) W.H.Press, S.A.Teukolsky, W.T.Vetterling and B.P.Flannery 著, 丹慶, 奥村, 佐藤, 小林 訳: ニューメリカルレシビ・イン・シー, 技術評論社, 282/338 (1993)
- 19) 佐久間, 小林: 実数値GAにおけるサンプリングバイアスを考慮した外挿的交叉EDX, 人工知能学会論文誌, 17-6, 699/707 (2002)
- 20) B.Sareni and L.Krähenbühl: Fitness Sharing and Niching Methods Revisited, IEEE Trans. on EC, 2-3, 97/106 (1998)
- 21) 佐藤, 小野, 小林: 遺伝的アルゴリズムにおける世代交代モデルの提案と評価, 人工知能学会論文誌, 12-5, 734/744 (1997)
- 22) M.G.-Schleuter: ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy, Proc. of 3rd ICGA, 422/427 (1989)
- 23) 染谷, 山村: 最適解の位置にロバストな実数値GAを実現するTroidal Search Space Conversionの提案, 人工知能学会論文誌, 16-3, 333/343 (2001)
- 24) O.Takahashi, H.Kita and S.Kobayashi: A Distance Dependent Alternation Model on Real-coded Genetic Algorithms, Proc. of SMC'99, 619/624 (1999)
- 25) 高橋, 木村, 小林: 交叉的突然変異による適応的近傍探索-騙しのある多峰性関数の最適化-, 人工知能学会論文誌, 16-2, 175/184 (2001)

《付 録》

A. 交叉 ENDX

ENDX¹⁰⁾は多数の親個体を利用し、主にそのうちの2個体の親を結ぶ直線に沿って正規分布に従って子個体を生成する交叉である。両親を結ぶ直線に直交する方向に関して残りの親個体を利用して摂動を加える。ENDXは以下の式に従って子個体を生成する。

$$\mathbf{c} = \mathbf{p} + \xi \mathbf{d} + \sum_{i=3}^m \eta_i \mathbf{p}'_i \quad (\text{A.1})$$

ただし,

$$\mathbf{p} = (\mathbf{p}_1 + \mathbf{p}_2)/2, \quad (\text{A.2})$$

$$\mathbf{d} = \mathbf{p}_2 - \mathbf{p}_1, \quad (\text{A.3})$$

$$\mathbf{p}'_i = \mathbf{p}_i - \frac{1}{m-2} \sum_{j=3}^m \mathbf{p}_j, \quad (\text{A.4})$$

$$\xi \sim N(0, \alpha^2), \quad (\text{A.5})$$

$$\eta_i \sim N(0, \beta^2). \quad (\text{A.6})$$

$\mathbf{p}_i$ は親個体の遺伝子ベクトル, $\mathbf{c}$ は子個体の遺伝子ベクトルである. α, β は定数パラメータであり, $\alpha = 0.434$, $\beta = 0.35/\sqrt{m-3}$ が推奨されている. また m は使用する親個体の数であり $m = n + 2$ が推奨されているが, 本論文では計算コスト削減の観点からすべての実験で $m = 12$ とした. ただし n は探索空間の次元数である.

B. 突然変異 NDM

NDM¹¹⁾ は ENDX に似た突然変異オペレータである. ENDX が 2 つの親個体 $\mathbf{p}_1, \mathbf{p}_2$ を結ぶ直線に沿って正規分布に従って子個体を生成する代わりに, NDM は親個体 $\mathbf{p}_1$ のまわりに正規分布に従って子個体を生成する. NDM のアルゴリズムは以下のとおりである.

$$\mathbf{c} = \mathbf{p}_1 + \sum_{i=2}^m \eta_i \mathbf{p}'_i. \quad (\text{B.1})$$

ただし,

$$\mathbf{p}'_i = \mathbf{p}_i - \frac{1}{m-1} \sum_{j=2}^m \mathbf{p}_j, \quad (\text{B.2})$$

$$\eta_i \sim N(0, \gamma^2). \quad (\text{B.3})$$

定数パラメータ $\gamma = 0.35/\sqrt{m-2}$ が推奨されている. また NDM が使用する親個体の数として $m = n + 2$ が推奨されているが, 本論文では ENDX と同じ理由から $m = 12$ とした.

C. 世代交代モデル MGG

MGG²¹⁾ は連続的な世代交代を行なうモデルである. ただし MGG は交叉のための世代交代モデルであるため, DIDC では突然変異も適用できるように変更が加えられている¹¹⁾. 改良された MGG のアルゴリズムを以下に示す.

[アルゴリズム: MGG]

- (1) 世代数を 0 に設定. 初期集団として n_p 個体をランダムに生成.
- (2) 集団から親個体として m 個体をランダムに非復元抽出. それぞれの個体を $\mathbf{p}_i$ とする ($i = 1, \dots, m$).
- (3) 選択した親個体に交叉または突然変異を適用することで, n_c 個体の子集団を生成する. なお ENDX における親個体 $\mathbf{p}_1, \mathbf{p}_2$ や NDM における $\mathbf{p}_1$ は残りの親個体とは異なり特別な意味を持つ (主要親個体と呼ぶ) が, 選択された m 個体の中からランダムに選択されるものとする.

(4) 使用された探索オペレータが交叉の場合は, 家族中の最良個体と, 家族中からランダムに選択された個体を主要親個体と交代させる. 突然変異の場合は家族中の最良個体をその主要親個体と交代させる. ただし家族とは主要親個体とその子個体のことである.

(5) 終了条件を満たしていれば終了. さもない場合は世代数を 1 増加させてステップ 2 へ.

[著者紹介]

木村 周平 (正会員)

1998 年京都大学大学院高分子化学専攻修士課程修了. 2001 年東京工業大学大学院知能システム科学専攻博士後期課程修了. 現在理化学研究所ゲノム科学総合研究センター研究員. 博士 (工学). 進化計算, バイオインフォマティクスなどの研究に従事. 人工知能学会, 日本分子生物学会などの会員.

高橋 治 (正会員)

1985 年東京工業大学制御工学科卒業. 同年, 東レ (株) 入社, 画像処理などの研究開発に従事. 2001 年東京工業大学大学院知能システム科学専攻博士後期修了. 同年, NEDO 養成技術者, 現在に至る. 進化計算とその応用研究に従事.

小林 重信 (正会員)

1969 年, 東京工業大学理工学部応用化学科卒業, 74 年, 同大学大学院理工学研究科経営工学専攻博士課程修了. 博士 (工学). 同年, 同大学工学部制御工学科助手. 81 年 8 月, 同大学大学院総合理工学研究科システム科学専攻助教授. 90 年 8 月, 同研究科生命科学専攻教授. 96 年 5 月より同研究科知能システム科学専攻に所属, 現在に至る. 創発システム論, 生物的適応システム, 進化計算, 強化学習の研究に従事.

小長谷 明彦 (正会員)

理化学研究所ゲノム科学総合研究センターゲノム情報科学グループプロジェクトディレクター, 東京工業大学大学院客員教授. 博士 (工学). 80 年東京工業大学理工学研究科情報科学専攻修士課程修了, 同年日本電気 (株) 入社, 96 年 10 月東京工業大学知能システム科学連携講座客員助教授, 97 年 10 月北陸先端科学技術大学院大学知識科学研究科教授, 2003 年 4 月より現職. バイオインフォマティクス, 高性能計算機技術, グリッド計算技術に興味をもつ.