数理解析研究所講究録
第1864巻 2013年 20-30

巣形成と役割分担を用いた最適化手法に関する一考察

広島修道大学商学部　阪井　節子 (Setsuko Sakai)
Faculty of Commercial Sciences, Hiroshima Shudo University
広島市立大学大学院 情報科学研究科　高濱　徹行 (Tetsuyuki Takahama)
Graduate School of Information Sciences, Hiroshima City University

1 はじめに

進化的アルゴリズム (Evolutionary Algorithm, EA) は，生物進化の過程をモデル化した最適化アルゴリズムの総称であり，遺伝的アルゴリズム (Genetic Algorithm, GA)，進化戦略 (Evolution Strategy, ES)，差分進化 (Differential Evolution, DE)[1, 2] など多くのアルゴリズムが提案されている．EA は，最適化の対象である目的関数の値だけを利用して解を求めることができる直接探索法であり，アルゴリズムの実装が容易であることから，様々な最適化問題を解くために利用されている．

しかし，EA は確率的な多点探索を行うため，比較的局所解に陥りにくいが，関数評価回数が多くなりがちである．近年，最適化問題が大規模化し，目的関数の評価コストが増大してきているため，目的関数の評価回数の削減は大きな課題となってきている．

本研究では，蟻の巣形成と役割分担に着目した最適化アルゴリズムを提案する．探索点から近接グラフを構成し目的関数の概形を把握する [3, 4, 5]．谷点およびその近傍の探索点が存在する領域は有望な探索領域である考えられるため，この領域を巣と見なす．谷点を女王蟻，谷近傍点を雄蟻とし，これらの蟻の近傍に新しい蟻，すなわち新しい探索点を生成する．それ以外の蟻は，採餌行動を行っている働き蟻とし，通常の探索により良い餌場を探して移動する．ただし，山点に到達した蟻は餌場の探索に失敗したと考え，帰巣本能により巣に帰る行動をとる．このようなアイデアに基づき，EA を効率化し，広い範囲の問題に対して，少ない関数評価回数で精度の高い解の探索を実現することが本研究の目的である．

2 最適化問題

本研究では，決定変数の上下限制約のみを有する次のような最適化問題 (P) を扱う．

$$(P)\ \text{minimize}\quad f(\boldsymbol{x}) \qquad \text{subject to}\ \ l_i \le x_i \le u_i,\ i = 1, \ldots, n \tag{1}$$

ここで，$\boldsymbol{x} = (x_1, \cdots, x_n)$ は n 次元決定変数ベクトル，$f(\boldsymbol{x})$ は目的関数であり，f は線形あるいは非線形の実数値関数である．l_i, u_i はそれぞれ，n 個の決定変数 x_i の下限値，上限値である．さらに，以下では全ての上下限制約を満足する領域を探索空間 ($\mathcal{S}$) と呼ぶことにする．

3 近接グラフ

3.1 定義

頂点集合 V と辺集合 E からなるグラフ G を $G(V, E)$ で表現する．近接グラフ (proximity graph) は，頂点集合 V の近傍構造を表現するグラフであり，最近傍グラフ (Nearest Neighborhood Graph)，Gabriel グラフ (Gabriel Graph, GG)[6]，相対近傍グラフ (Relative Neighborhood Graph, RNG)[7]，β skeleton[8]，競合ヘブ則によるグラフ [9] などが提案されている．近接グラフでは，任意の 2 頂点 $v_i, v_j \in V$ が近傍条件を満足するときのみ辺 $(v_i, v_j) \in E$ となる．

Gabriel グラフでは，v_i と v_j を結ぶ線分を直径とする超球内に他の頂点が含まれていなければ，v_i と v_j が近傍条件を満足する．Gabriel グラフ $GG(V, E)$ は，以下のように定義できる．

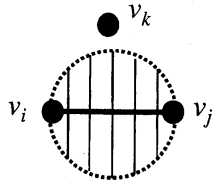
$$(v_i, v_j) \in E \iff HS\left(\frac{v_i + v_j}{2}, \frac{||v_i - v_j||}{2}\right) \cap V = \phi \tag{2}$$

ここで，$HS(v,r)$ は頂点 v を中心とする半径 r の超球内の領域，すなわち

$$HS(v,r) = \{x \mid ||x - v|| < r\} \tag{3}$$

である．

この様子を図 1 に示す．任意の頂点 v_k が超球内に存在しない場合にのみ頂点を接続し，そうでない場合は接続しない．



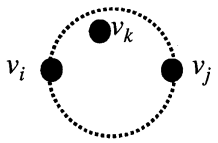
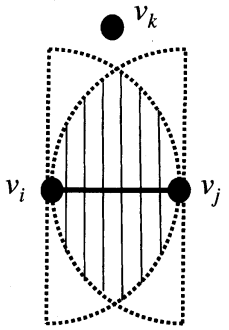


図 1: Gabriel グラフにおける辺の判定



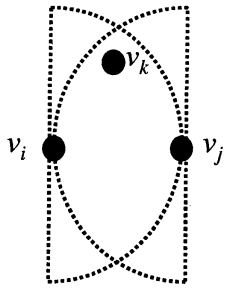


図 2: 相対近傍グラフにおける辺の判定

相対近傍グラフでは，頂点 v_i および v_j を中心とする半径 $||v_i - v_j||$ の 2 つの超球の共通集合内に他の頂点が含まれていなければ近傍条件を満足する．相対近傍グラフは Gabriel グラフの部分グラフである．相対近傍グラフ $RNG(V,E)$ は以下のように定義できる．

$$(v_i, v_j) \in E \iff HS(v_i, ||v_i - v_j||) \cap HS(v_j, ||v_i - v_j||) \cap V = \phi \tag{4}$$

この様子を図 2 に示す．

3.2 近接グラフのアルゴリズム

近接グラフを決定するアルゴリズムは以下のようになる．

1. 全ての頂点対について，v_i, v_j 間の距離 $d(v_i, v_j), i, j = 1, 2, \cdots, n$ を求める．
2. 全ての頂点対 v_i, v_j について，
   (a) ある頂点 $v_k (k \neq i, k \neq j)$ に対して，
   Gabriel グラフの場合は，$d(v_i, v_k)^2 + d(v_j, v_k)^2 < d(v_i, v_j)^2$ ならば，v_i, v_j は近傍関係にない．
   相対近傍グラフの場合は，$d(v_i, v_k) < d(v_i, v_j)$ かつ $d(v_j, v_k) < d(v_i, v_j)$ ならば，v_i, v_j は近傍関係にない．
   (b) 上記を満たす v_k が存在しなければ，辺 (v_i, v_j) を生成する．

なお，アルゴリズムの計算量は，$O(|V|^3)$ である．

4 巣形成と役割分担に基づく最適化アルゴリズム

本研究では，Differential Evolution (DE) に巣形成と役割分担の考えを導入した最適化アルゴリズムである NRDE(Differential Evolution with Nest-building and Role-sharing) を提案する．

4.1 Differential Evolution

Differential evolution (DE) は，Storn and Price[1, 2] によって提案された進化的アルゴリズム (Evolutionary Algorithm) の 1 つである．DE は解集団を用いた多点探索を行う確率的直接探索法であり，非線形問題，微分不可能な問題，非凸問題，多峰性問題など，様々な最適化問題に適用されてきており，高速で頑健なアルゴリズムであることが示されている．

DE では，探索空間中にランダムに初期個体を生成し，初期集団を構成する．各個体は決定ベクトルに対応し，n 個の決定変数を遺伝子として持つ．各世代において，全ての個体を親として選択する．各親に対して，次のような処理が行われる．突然変異のために，選択された親を除く個体群から互いに異なる $1+2\,num$ 個の個体を選択する．最初の個体が基本ベクトル (base vector) となり，残りの個体対が num 個の差分ベクトル (difference vector) となる．差分ベクトルはスケーリングファクタ F(scaling factor) が乗算され基本ベクトルに加算され，変異ベクトル (mutant vector) が得られる．変異ベクトルと親が交叉し，交叉率 CR(crossover rate) により指定された確率で親の遺伝子をベクトルの要素で置換することにより，子のベクトル (trial vector) が生成される．最後に，生存者選択として，子が親よりも良ければ，親を子で置換する．

DE には幾つかの形式が提案されており，DE/best/1/bin や DE/rand/1/exp などがよく知られている．これらは，DE/*base*/*num*/*cross* という記法で表現される．"*base*" は基本ベクトルとなる親の選択方法を指定する．例えば，DE/rand/*num*/*cross* は基本ベクトルのための親を集団からランダムに選択し，DE/best/*num*/*cross* は集団の最良個体を選択する．"*num*" は基本ベクトルを変異させるための差分ベクトルの個数を指定する．"*cross*" は子を生成するために使用する交叉方法を指定する．例えば，DE/*base*/*num*/bin は一定の確率で遺伝子を交換する交叉 (binomial crossover) を用い，DE/*base*/*num*/exp は，指数関数的に減少する確率で遺伝子を交換する交叉 (exponential crossover) を用いる．

本研究では，差分ベクトル数を 1 ($num = 1$) とした DE/rand/1/exp を用いる．以下に DE/rand/1/exp の擬似コードを示す．

```
DE/rand/1/exp()
{
// Initialize a population
 P=N individuals generated randomly in S;
 for(t=1; FE ≤ FE_max; t++) {
  for(i=1; i ≤ N; i++) {
// DE operation
     x^p1=Randomly selected from P(p1 ≠ i);
     x^p2=Randomly selected from P(p2 ∉ {i,p1});
     x^p3=Randomly selected from P
                              (p3 ∉ {i,p1,p2});
     x'=x^p1+F(x^p2 − x^p3);
     x^child=trial vector is generated from
           x^i and x' by exponential crossover;
// Survivor selection
     if(f(x^child) ≤ f(x^i))  z^i=x^child;
     else                     z^i=x^i;
     FE=FE+1;
  }
  P={z^i, i = 1,2,···,N};
 }
}
```

4.2 山谷構造の推定と山谷判定

本研究では，山と谷を判定するために，各点 $\boldsymbol{x}_i$ に対して山度 $\boldsymbol{x}_i.hill$ と谷度 $\boldsymbol{x}_i.valley$ を付与する．近傍グラフの全ての辺について，その辺を構成する 2 点のうち，評価値の良い点の谷度を 1 増加させ，評価値の悪い点の山度を 1 増加させる．

本研究では，各点の谷度と山度を用いて，3 種類の点，谷点，山点，谷近傍点を定義し，各点 $\boldsymbol{x}_i$ の山谷判定を行う．

谷点 近傍により悪い点が一つ以上存在し，より良い点が存在しない点である．すなわち，谷点は $\boldsymbol{x}_i$.valley> 0 かつ $\boldsymbol{x}_i$.hill$= 0$ の点である．

山点 近傍により良い点が一つ以上存在し，より悪い点が存在しない点である．すなわち，$\boldsymbol{x}_i$.hill> 0 かつ $\boldsymbol{x}_i$.valley$= 0$ の点である．

谷近傍点 谷点に隣接する点，すなわち 1 つの辺を共有する点でかつ，山点でない点である．

4.3 アルゴリズム

以下にアルゴリズムの概要を示す.

1. 初期化
   探索領域内に点をランダムに生成し，初期探索点集合を構成する.
2. 近傍グラフの構成
   点間の距離を求める．近傍関係を求め，近傍グラフを構成する.
3. 山谷判定
   各点に山度と谷度を付与し，山谷判定を行う.
4. 探索点の生成
   女王蟻 (谷点) および雄蟻 (谷近傍点) は局所探索を行うため，局所的な DE 操作を行う．谷点を中心とする探索を行うために，スケーリングファクタ F に小さな値，交叉率 CR に大きな値を設定する．山点では，標準的な探索をあきらめ，最良の谷点へ向かって移動する．すなわち帰巣行動 (中域探索) を行う．このために，F に大きな値，CR は [0,1] の乱数で値を設定する．それ以外の点については，標準的な探索を続行，すなわち通常のパラメータ値 F と CR による DE 操作を行う.
5. 生存者選択
   元の点が生成された探索点より良好でなければ，探索点と置換する．2. に戻る.

4.4 探索点の生成

本研究では，山谷判定による各親ベクトル $\boldsymbol{x}_i$ の属性に応じて，以下のように基本ベクトルの選択戦略およびスケーリングファクター F と交叉率 CR の制御ルールを採用することにより，近傍探索，中域探索と大域探索を実現する.

谷点の場合
: 谷点 $\boldsymbol{x}_i$ の周辺で局所探索を行うため，基本ベクトルを谷点 $\boldsymbol{x}_i$ とし，$F = 0.3$，$CR = 1.0$ とする.

谷近傍点の場合
: 谷点の場合と同様に，谷点の周辺で局所探索を行うため，基本ベクトルを近傍の谷点と $\boldsymbol{x}_i$ の中点とし，$F = 0.4$，$CR = 1 - \frac{1}{n}$ とする.

山点の場合
: 山点である蟻は，標準的な探索をあきらめ帰巣行動をとる．そのため，基本ベクトルを集団内の最良解 (最良の谷点) とし，$F = 0.9$，CR は区間 $[0, 1]$ 上の一様乱数とする.

それ以外の場合
: 標準的探索を続行する．そのため，基本ベクトルは $\boldsymbol{x}_i$ を除く集団からランダムに選択し，F と CR は通常の DE で標準的に用いられる $(0.7, 0.9)$ とする.

4.5 擬似コード

以下に DE/rand/1/exp の擬似コードを参考に，NRDE の擬似コードを示す.

```
NGDE()
{
  P=Generate N individuals {x_i} randomly;
  Evaluate x_i, i = 1,2,···,N;
  for(t=1; t < T_max; t++) {
  // 距離などの初期化
    for(i=1; i ≤ N; i++) {
      x_i.num=x_i.valley=x_i.hill=0;
      for(j=i+1; j ≤ N; j++)
        d_ij=d_ji=x_i と x_j 間の距離 d(x_i,x_j);
    }
  // 近傍グラフの構成
    for(i=1; i ≤ N; i++) {
      for(j=i+1; j ≤ N; j++) {
        for(k=1; k ≤ N; k++) {
          if(k!=i && k!=j &&
             d_ik^2 + d_jk^2 < d_ij^2 // GG
             d_ik < d_ij && d_jk < d_ij // RNG
            ) break;
        }
        if(k>N) {
      // 山谷判定: 辺 (x_i, x_j) の生成
          if(f(x_i) < f(x_j)) {
            x_i.valley++; x_j.hill++;
          }
          else if(f(x_i) > f(x_j)) {
            x_j.valley++; x_i.hill++;
          }
      }
    }
  }
    全ての個体について山谷判定を行う;
    for(i=1; i ≤ N; i++) {
      CurrentToQueen=0;
      switch(x_i の種類) {
        case 谷点:
            p_1=i; F'=0.3; CR'=1;
            break;
        case 谷近傍点:
            p_1=i_valley; F'=0.4; CR'=1-1/n;
            CurrentToQueen=1;
            break;
        case 山点:
            p_1=best;
            F'=0.9; CR'=[0,1] の乱数;
            break;
        default:
            p_1=select randomly in [1,N]\{i};
            F'=F; CR'=CR;
        }
      // 探索点の生成
        p_2=select randomly in [1,N]\{i,p_1}
        p_3=select randomly in [1,N]\{i,p_1,p_2}
        x_i^new=x_i ∈ P;
        j=select randomly from [1,n];
        k=1;
        do {
          x_ij^new=x_p1,j+F'(x_p2,j - x_p3,j);
          if(CurrentToQueen)
            x_ij^new+=0.5(x_i,j - x_p1,j);
          j=(j+1)%n;
          k++;
        } while(k ≤ n && u(0,1) < CR');
      // 生存者選択
        if(f(x^new) ≤ f(x_i)) x_i=x^new;
      }
    }
}
```

ただし，$\boldsymbol{x}_i$.hill, $\boldsymbol{x}_i$.valley はそれぞれ点 $\boldsymbol{x}_i$ より良い近傍頂点数，悪い近傍頂点数であり，$\boldsymbol{x}_{i_{valley}}$ は $\boldsymbol{x}_i$ の隣接谷点，$\boldsymbol{x}_{best}$ は集団内の最良点である．

5 実験

5.1 テスト問題

本実験では，変数間依存性，悪スケール性，および大谷構造を有する問題を用いる [10]．変数間依存性の強い問題として，稜構造を有する問題を用いる．悪スケール性のある問題として，稜構造でかつ変数によりスケールが異なる問題を用いる．大谷構造とは，微視的に見れば局所解となる多数の小さな谷が存在するが，巨視的に見れば大きな谷が一つだけ存在し，その谷が最適解となっている構造であり，Rastrigin 関数がその典型例である．

以下に，関数とその初期化領域を示す．なお，n は次元数を表している．

- f_1: Sphere 関数

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} x_i^2, \quad -5.12 \leq x_i \leq 5.12 \tag{5}$$

単峰性の関数で，点 $(0, 0, \cdots, 0)$ で最小値 0 をとる．

- f_2: Rosenbrock 関数

$$f(\boldsymbol{x}) = \sum_{i=2}^{n} \{100(x_1 - x_i^2)^2 + (x_i - 1)^2\}, \quad -2.048 \leq x_i \leq 2.048 \tag{6}$$

単峰性の稜構造を有する関数で，点 $(1, 1, \cdots, 1)$ で最小値 0 をとる．

- f_3: ill-scaled Rosenbrock 関数

$$f(\boldsymbol{x}) = \sum_{i=2}^{n} \{100(x_1 - (ix_i)^2)^2 + (ix_i - 1)^2\}, \quad -2.048/i \leq x_i \leq 2.048/i \tag{7}$$

単峰性の稜構造を有する関数で，点 $(1, \frac{1}{2}, \cdots, \frac{1}{n})$ で最小値 0 をとる．

- f_4: Rastrigin 関数

$$f(\boldsymbol{x}) = 10n + \sum_{i=1}^{n} \{x_i^2 - 10\cos(2\pi x_i)\}, \quad -5.12 \leq x_i \leq 5.12 \tag{8}$$

多峰性の大谷構造を有する関数で，点 $(0, 0, \cdots, 0)$ で最小値 0 をとる．

図 5 に $n = 2$ のときの関数 f_1, f_2, f_4 のグラフを示す．また，表 1 に各関数の特徴を示した [10].

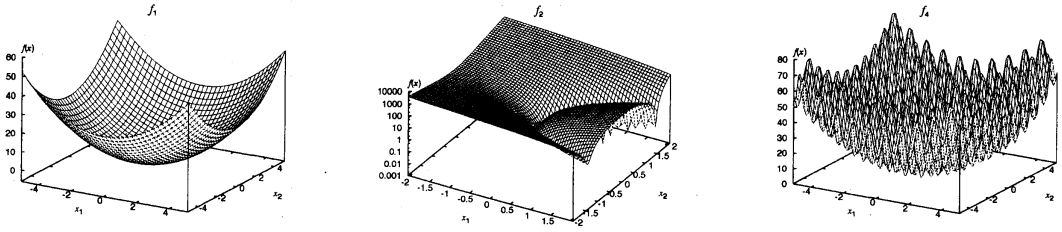


図 5: 関数 f_1,f_2,f_4 のグラフ

表 1: テスト問題の特徴

関数	変数間依存性	悪スケール性	大谷構造
f_1	なし	なし	なし
f_2	強い	なし	なし
f_3	強い	あり	なし
f_4	なし	なし	あり

5.2 実験条件

次元数 $n = 30$ に設定し，f_1 から f_4 の関数を最適化する．rand 戦略を用いる標準の DE，近接グラフとして相対近傍グラフ (RNG) あるいは Gabriel グラフ (GG) を利用した NRDE の 3 つのアルゴリズムの性

能を比較する．DE の設定は，個体数 $N = 50$, F と CR の組合せを $(0.7, 0.9)$ とし，各関数について 30 回の試行を行い，結果を考察する．個体数は $2n$ から $10n$ 程度が良いとされているが，効率性を重視し少し小さい値とした．なお，試行打ち切り条件は，文献 [10] と同様に，(1) 評価値が 1.0×10^{-7} 以下に達した，(2) 探索打ち切り評価回数が f_1 および f_2 は $2n \times 10^5$，f_3 は $5n \times 10^5$，f_4 は $3n \times 10^5$ に達した，のいずれかの条件を満足する場合とした．

5.3 実験結果

実験結果を表 2 に示す．許容精度を満足した回数を "$< 1e{-}7$"，実際に関数を評価した回数とその標準偏差を eval，そのうち子ベクトルが良かった回数を succ，悪かった回数を fail, 成功率を rate に示した．

表 2: 実験結果

function	algorithm	$< 1e^{-7}$	eval	success	fail	rate(%)
f_1	rand(0.7,0.9)	30	57,865.07± 848.40	11744.00	46070.07	20.31
	RNG(0.7,0.9)	30	32,547.03 ± 621.45	10642.93	21853.10	32.75
	GG(0.7,0.9)	30	**26,750.07**±3071.48	15174.93	11524.13	56.84
f_2	rand(0.7,0.9)	30	558,147.93±14635.54	19614.43	538482.50	3.51
	RNG(0.7,0.9)	30	**98,521.40**± 3551.71	14807.93	83662.47	15.04
	GG(0.7,0.9)	2	156,439.50± 8081.50	59148.00	97240.50	37.82
f_3	rand(0.7,0.9)	30	560,235.00±15001.30	19588.47	540595.53	3.50
	RNG(0.7,0.9)	30	**84,189.63**± 3833.30	16230.03	67908.60	19.29
	GG(0.7,0.9)	10	177,631.90±65173.52	74575.20	103005.70	42.00
f_4	rand(0.7,0.9)	30	162,201.80± 4280.73	14985.87	147164.93	9.24
	RNG(0.7,0.9)	30	**132,934.70**± 4812.70	14178.93	118704.77	10.67
	GG(0.7,0.9)	23	222,500.83±19602.14	16961.91	205487.91	7.63

f_1 については，NRDE(GG) が最も効率よく近似解を探索できており，次に NRDE(RNG) の効率が優れている．f_2, f_3 については，NRDE(RNG) が最も効率よく近似解を探索できている．次に NRDE(GG) の効率が良いが，f_2 では 28 回の試行，f_3 では 20 試行で近似解の発見に失敗しているため，実質的には DE(rand) の方が NRDE(GG) より効率が優れていると考えられる．f_4 については，NRDE(RNG) が最も効率よく近似解を探索できており，次に DE(rand) の効率が優れている．NRDE(GG) では 7 試行で近似解の発見に失敗している．以上のことから，安定的な探索の観点からは，NRDE(RNG) がもっとも優れており，次に DE(rand) となる．NRDE(GG) は問題に依存した不安定さがある．また，近似解の発見までに要する関数評価回数について NRDE(RNG) と DE(rand) を比較すると，f_1 については約 44%，f_2 については約 82%，f_3 については約 85%，f_4 については約 18%の削減に成功している．以上のことから，探索効率の点からも NRDE(RNG) が最も優れている．

各問題における最良個体の関数値について，その平均値の変化を図 6 から図 9 に示す．横軸は世代数，縦軸が関数値である．

f_1 については，NRDE(GG) が最も効率的に探索を行っており，NRDE(RNG) がほぼ同程度の効率である．DE(rand) はかなり遅い．f_2，f_3 については，NRDE(RNG) が非常に効率的に探索し，DE(rand) は探索が遅く，NRDE(GG) は最大評価回数までに満足できる解を探索できなかった．f_4 についても，NRDE(RNG) が最も効率的に探索を行っており，DE(rand) がやや劣った効率である．NRDE(GG) は満足できる解を発見できない場合があった．

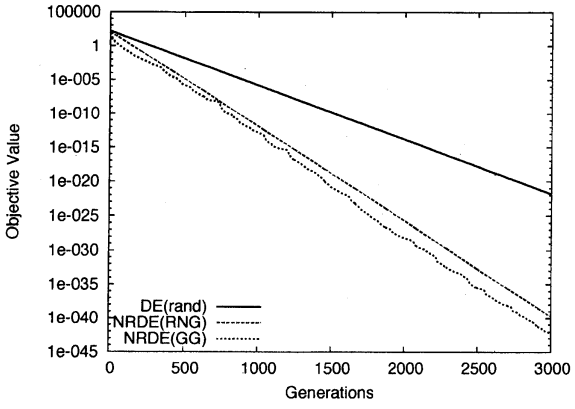


図 6: f_1 における関数値の変化

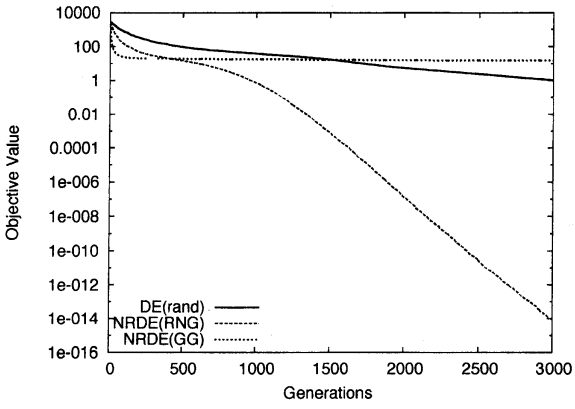


図 7: f_2 における関数値の変化

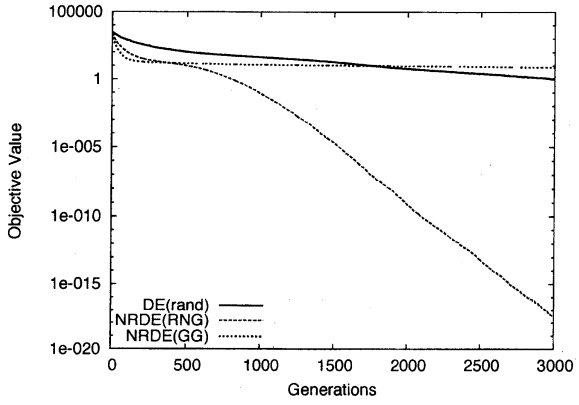


図 8: f_3 における関数値の変化

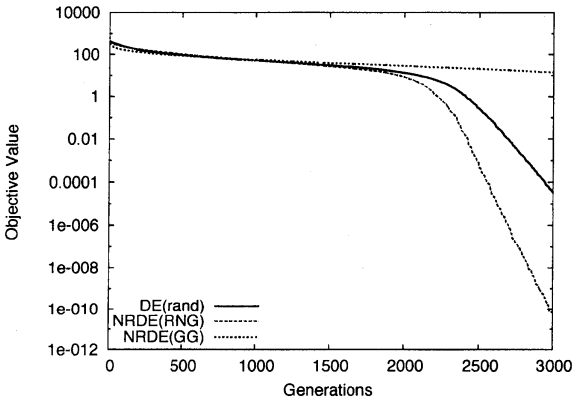


図 9: f_4 における関数値の変化

谷 (valley)，谷近傍 (near valley)，山 (hill) の点の数の変化 (左軸参照) と最良値 (best, 右軸参照) の変化について，NRDE(RNG) の場合を図 10 から図 13 に，NRDE(GG) の場合を図 14 から図 17 に示す．

NRDE(RNG) において，f_1 については，谷点は定常的に約 2 個であり，谷近傍点は定常的に 7 個から 8 個程度の間で増減を繰り返している．山点は定常的に 30 個程度である．すなわち，巣の近傍での近傍探索は 10 個体程度，巣への帰巣行動は 30 個体程度，通常の DE による大域探索を 10 個体程度が定常的に行っていると考えられる．f_2 については，初期は谷点が 3〜5 個体まで減少し，その後増加し，中盤以降は定常的に 10 個体程度となっている。谷近傍点も 5〜8 個体程度まで減少し，その後は谷点と同様の変化を見せている．山点は初期に 30 個体程度まで増加し，その後減少し，再度やや増加し，中盤以降は定常的に 25 個体程度となっている．最良値の変化とあわせると，探索の中盤までは，谷点として局所解である $(0,0,\cdots,0)$ しか発見できておらず，その後最適解である $(1,1,\cdots,1)$ も発見して，その結果谷点及び谷近傍点が増加したと考えられる．最良値の最適値への収束もこの変化に同期していることが分かる．

f_3 については，初期は谷点が 6〜9 個体，谷近傍が 9〜11 個程度で，その後増加し，中盤以降は定常的にそれぞれ 14 個，11 個体程度となっている。山点は初期に 18〜20 個であるが，その後定常的に 20 個体程度となっている．最良値の変化とあわせると，f_2 と同様に，一度 $(0,0,\cdots,0)$ 付近に収束し，そこから狭小な谷を通って $(1,\frac{1}{2},\cdots,\frac{1}{n})$ に向かって移動するという探索方法がとられていると考えられる．f_4 については，序盤は谷点が 4 から 14 個，谷近傍点が 8 から 18 個へ増加し，山点は 24 から 10 個へ減少している．中盤は定常的に谷点が 13 個程度，谷近傍点が 19 個程度，山点 8 個程度である．終盤には谷点は 3 個程度，谷近傍点は 7 個程度まで減少し，谷点は 30 個程度まで増加増加している．最良値の変化とあわせると，f_4 は

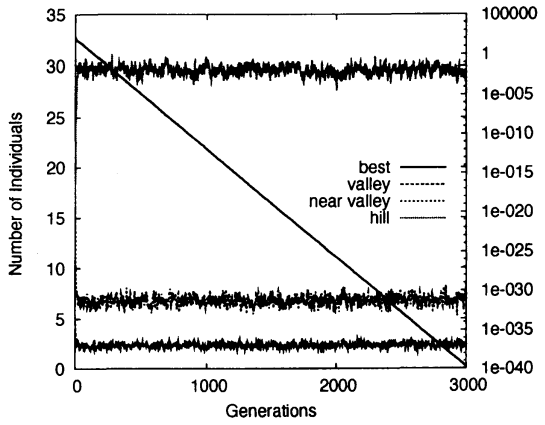


図 10: f_1 における山谷判定 (NRDE/RNG)

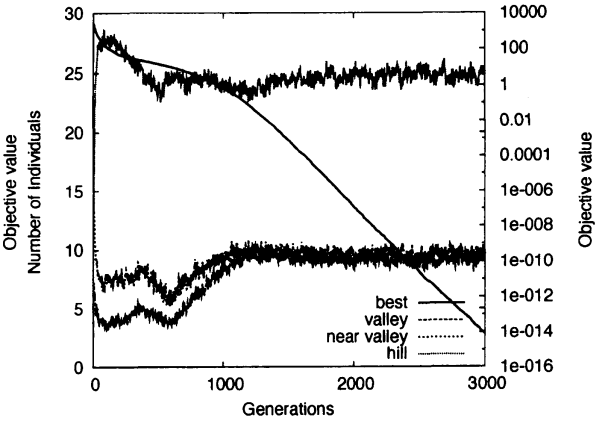


図 11: f_2 における山谷判定 (NRDE/RNG)

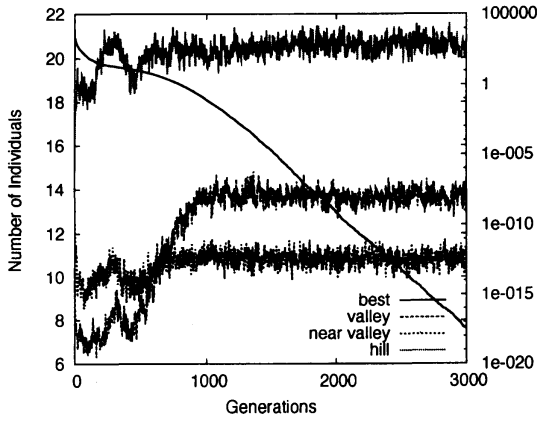


図 12: f_3 における山谷判定 (NRDE/RNG)

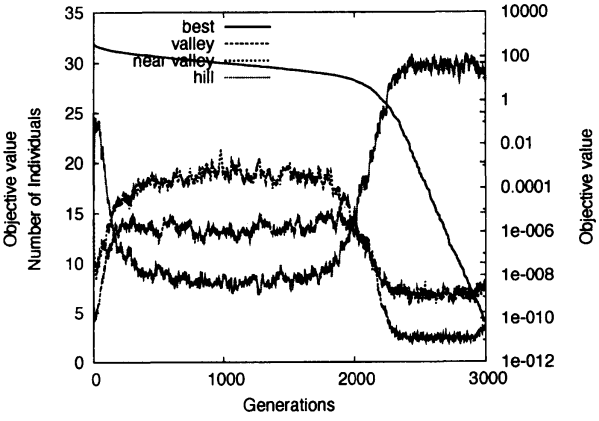


図 13: f_4 における山谷判定 (NRDE/RNG)

巨視的には谷がただ一つだけ存在するため，序盤は探索空間内にランダムに生成した探索点は大域的最適値方向へ探索が行われる．中盤は f_4 の多数の小さな谷にとらえられるため谷点および谷近傍点が増加する．探索が大域的最適解の周辺まで進むと，大域的最適解である谷点の周辺は局所的には f_1 と同様な単峰性あるため，終盤では谷点と谷近傍点が再度減少し，最良値が急速に最適値に収束している．

NRDE(GG) の f_1 から f_4 に共通した特徴として，NRDE(RNG) に比して谷近傍点の個数が多いことがあげられる．RNG は GG の部分グラフであるため，GG は RNG に比して完全グラフに近いグラフである．したがって 1 個の頂点に連結する辺の数が多くなり，谷点に連結する点である谷近傍点が多くなっている．f_1 については，谷点は定常的に約 2 個であり，谷近傍点は定常的に 20 個から 27 個程度の間で増減を繰り返している．山点は定常的に 4 個程度である．すなわち，巣の近傍での近傍探索は 30 個体程度，巣への帰巣行動は 4 個体程度，通常の DE による大域探索を 15 個体程度が定常的に行っていると考えられる．f_2 および f_3 については，谷点が十分に減少していない．f_4 については，谷点が定常的に 1 個程度となっている．最良値が最適値に収束していないことから，局所解に完全に収束してしまっていると考えられる．このように探索に失敗した原因は，GG が完全グラフに近い密なグラフであるために，f_1 のような単峰性関数に対しては探索性能を有するが，変数依存性や悪スケール性を有する関数 f_2 や f_3，大谷構造を有する f_4 では谷を十分とらえられなかったためと考えられる．

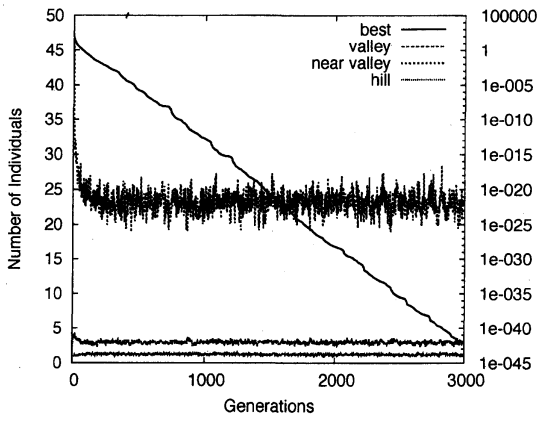


図 14: f_1 における山谷判定 (NRDE/GG)

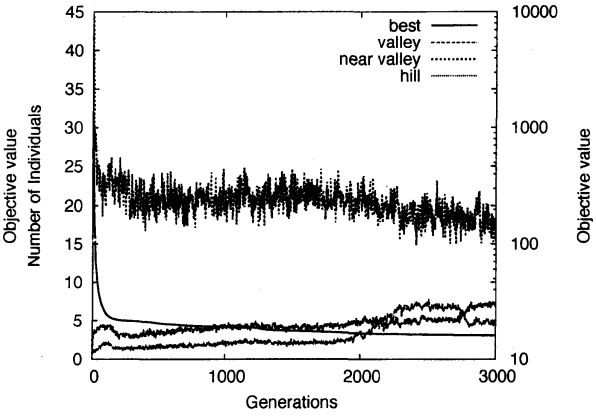


図 15: f_2 における山谷判定 (NRDE/GG)

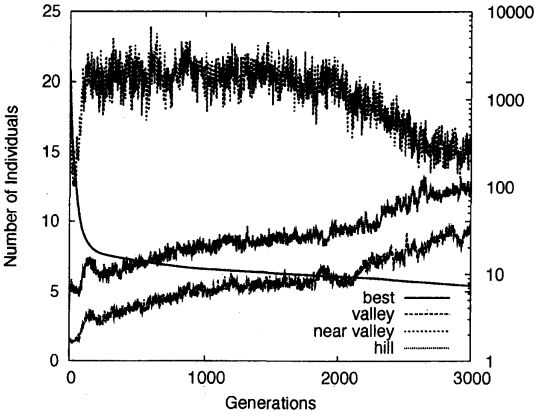


図 16: f_3 における山谷判定 (NRDE/GG)

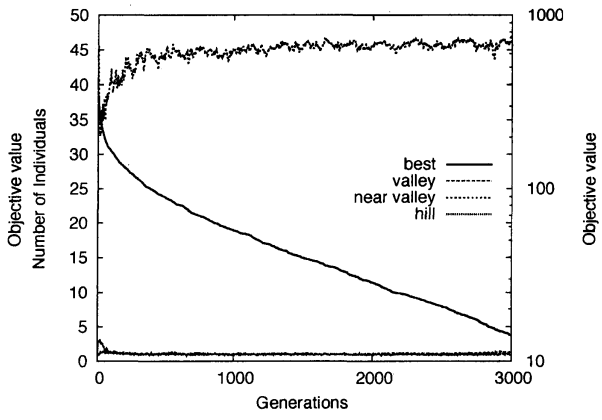


図 17: f_4 における山谷判定 (NRDE/GG)

6 おわりに

本研究では，集団的最適化アルゴリズムにおいて，探索効率を向上する方法として，巣形成と役割分担に基づく最適化アルゴリズムを提案した．提案した巣形成と役割分担に基づく最適化アルゴリズムは，探索点による近接グラフに基づく山谷構造の推定と山谷判定を行い，探索点の近傍における山谷構造に基づいて探索モードを変更することで，探索効率を向上する方法である．山谷構造の推定では近接グラフとしてガブリエルグラフと相対近傍グラフを用い，探索点を谷点，山点，谷近傍点，その他の点の 4 種類に分類した．巣形成と役割分担としては，谷点 (女王蟻) と隣接する谷近傍点 (雄蟻) が巣を形成し近傍探索を行い，山点は最良の巣へ帰巣する中域探索を行い，その他の点は通常の大域探索を行うこととした．本研究では探索アルゴリズムとして DE を採用した巣形成と役割分担に基づく最適化アルゴリズム NRDE を提案し，ガブリエルグラフによる NRDE(NRDE(GG))，相対近傍グラフによる NRDE(NRDE(RNG)) 及び標準的 DE(DE/rand) を比較することにより，単峰性関数に対しては NRDE(GG) が，稜構造および多峰性関数については NRDE(RNG) が他の方法と比較して，効率性の高い方法であることを示した．特に，NRDE(RNG) はいずれの問題に対しても安定的最適化アルゴリズムであることが示された．これにより，NRDE を利用して探索性能を向上できることを示した．

今後は，近接グラフとして競合ヘブ則によるグラフを用いること，局所探索と大域探索の実現方法を検討すること，self-adaptive のようにパラメータを動的に制御する手法と比較を行う，PSO などの集団的最適化アルゴリズムに適用することなどを予定している．

謝辞　この研究の一部は，日本学術振興会科学研究費補助金 基盤研究 (c) (No. 22510166, 24500177) および広島市立大学特定研究費（一般研究）の援助を受けた．

参考文献

[1] R. Storn and K. Price: "Minimizing the real functions of the ICEC'96 contest by differential evolution", Proc. of the International Conference on Evolutionary Computation, pp. 842–844 (1996).

[2] R. Storn and K. Price: "Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces", Journal of Global Optimization, **11**, pp. 341–359 (1997).

[3] 阪井, 高濱：“多次元空間における近傍構造を利用した最適化アルゴリズムに関する一検討”, 不確実・不確定性下での意思決定過程, 数理解析研究所講究録 1682, pp. 184–192 (2010).

[4] 高濱, 阪井：“競合ヘブ則によるグラフ生成に基づく種分化型 differential evolution の提案”, 第 22 回インテリジェント・システム・シンポジウム講演論文集.

[5] T. Takahama and S. Sakai: "Differential evolution with graph-based speciation by competitive hebbian rules", Proc. of the Sixth International Conference on Genetic and Evolutionary Computing (ICGEC2012), pp. 445–448 (2012).

[6] K. R. Gabriel and R. R. Sokal: "A new statistical approach to geographic variation analysis", Systematic Zoology, **18**, pp. 259–270 (1969).

[7] G. T. Toussaint: "The relative neighborhood graph of a finite planar set", Pattern Recognition, **12**, 4, pp. 261–268 (1980).

[8] D. G. Kirkpatrick and J. D. Radke: "A framework for computational morphology", Computational geometry (Ed. by G. Toussaint), North-Holland, pp. 217–248 (1985).

[9] T. Martinetz and K. Schulten: "Topology representing networks", Neural Networks, **7**, 3, pp. 507–522 (1994).

[10] 田中, 土谷, 佐久間, 小野, 小林：“Saving MGG: 実数値 GA/MGG における適応度評価回数の削減”, 人工知能学会論文誌, **21**, 6, pp. 547–555 (2006).