数理解析研究所講究録
第1548巻 2007年 61-70

最適化手法における関数評価回数の削減手法
—ポテンシャルモデルに基づく比較推定法の提案—

広島修道大学商学部　阪井　節子 (Setsuko Sakai)
Faculty of Commercial Science, Hiroshima Shudo University
広島市立大学情報科学部　高濱　徹行 (Tetsuyuki Takahama)
Faculty of Information Sciences, Hiroshima City University

1　はじめに

最適化アルゴリズムは，与えられた領域において最小値 (最大値) をとる解を探索するためのアルゴリズムである．一般に最適化アルゴリズムの良さを決定することは困難であるが，本研究では以下に示す2つの点に着目する．

- 網羅性: 局所解に陥らず大域最適解を探すこと．特に多峰性の問題において局所解を避けるには網羅性が重要である．
- 効率性: 目的関数が大規模でその評価に時間がかかる問題において，目的関数の評価回数を減らすことは非常に重要である．例えば，実験やシミュレーションによって目的関数値が与えられる場合にはできる限り評価回数を少なくする必要がある．

網羅性と効率性に優れたアルゴリズムとして，集団的降下法に基づくアルゴリズムが注目されている．集団的降下法とは，複数の解の集合により集団を形成し，集団から得られる情報に基き，集団の各要素に対して順次新しい解を生成し，新しい解が良ければ古い解と置換するという方法である．

集団的降下法の代表として，Differential Evolution (DE) と Particle Swarm Optimization (PSO) がある．DE では，個体により集団を形成し，各個体から交叉と突然変異により新しい個体を生成し，新しい個体が良ければ古い個体と置換する [1, 2, 3, 4]．PSO では，解に対応する位置の情報を持つエージェントにより集団を形成し，各エージェントの現在位置とエージェントの最良位置と集団の最良位置により新しい位置を生成し，その位置が最良位置より良ければ古い最良位置と置換する [5, 6, 7, 8, 9, 10, 11]．これらの方法は，集団の各要素が降下法により個別に最適化されるため，集団が急速に特定の解に集中することが少なく，網羅性の高い探索が行われる．また，各要素が新しい解を生成する際に集団の情報を利用できるため，一つの点による降下法と比較すると局所解に陥りにくい効率の良い探索が行われる．

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．このような問題の例としては，構造設計最適化 (structural design optimization) 問題がある．計算流体力学 (computational fluid dynamics, CFD) シミュレーションを要する空力設計最適化 (aerodynamic design optimization) では，CFD シミュレーションのために1回の計算に数10時間かかる場合もある．このため，目的関数の評価回数を抑えるべく，さらに効率的な最適化アルゴリズムの開発が期待されている [12, 13, 14]．

本研究では，集団的降下法の効率をさらに向上させるために，比較推定法を提案する．比較推定法では，近似モデルにより解の良さを推定し，生成された新しい解が古い解より良いと推定された場合にのみ新しい解を評価する。その結果，新しい解が古い解より良ければ置換することにより，目的関数の評価回数を抑えることを目指す．

本論文では，近似モデルとしてポテンシャルに基づくモデルを用いた比較推定法 (以後これを，ポテンシャル法と呼ぶ) を，集団的降下法の一つである DE に対して組み込む。ポテンシャル法を組み込んだ方法を従来の DE と比較することにより，関数評価回数が削減でき，効率性が向上することを数値実験により示す．

本論文は次のように構成されている．2. で最適化問題を定義し，集団的降下法とその改良点について説明する．3. でポテンシャル法を提案する．4. でポテンシャル法を DE に組み込んだ potential DE 法を提案する．5. で実験結果を示す．6. はまとめである．

2 最適化問題と集団的降下法

2.1 最適化問題

一般的な最適化問題 (P) は，不等式制約，等式制約，上下限制約を有しており，以下のように定義できる．

$$\begin{aligned} (P)\ \text{minimize} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & g_j(\boldsymbol{x}) \leq 0,\ j = 1, \ldots, q \\ & h_j(\boldsymbol{x}) = 0,\ j = q+1, \ldots, m \\ & l_i \leq x_i \leq u_i,\ i = 1, \ldots, n \end{aligned} \tag{1}$$

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，$g_j(\boldsymbol{x}) \leq 0$ は q 個の不等式制約，$h_j(\boldsymbol{x}) = 0$ は $m - q$ 個の等式制約であり，f, g_j, h_j 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．

目的関数および制約条件がともに線形の場合が線形計画問題，その他の場合が非線形計画問題である．本研究では，不等式制約および等式制約のない非線形計画問題を対象とする．

2.2 集団的降下法

まず，DE や PSO などのように解集団による最適化の際に降下法を利用した最適化法である集団的降下法について説明する．集団的降下法は一般に以下のように記述できる．

1. 初期化: 集団に属する解をランダムに発生する
2. 評価: 全ての解を評価する
3. 終了判定: 終了条件を満足すれば終了する
4. 各解に対して，
   (a) 生成: 各解と集団の情報に基づき新しい解を生成する
   (b) 評価: 新しい解を評価する
   (c) 更新: 新しい解が古い解より良ければ，古い解を新しい解で置換する
5. 3. へ戻る

2.3 比較推定法

本研究では，関数評価回数を削減する手法を提案するが，できる限り汎用的な手法とするために，導入が容易であり，それぞれのアルゴリズムの特徴を生かすことに配慮した．このため，前節 (4b) および (4c) の部分のみを変更し，以下のように，新しい解を評価する前に，近似モデルに基づき，古い解より良いと推定された解だけを評価するように変更する．

1. `if` 新しい解が古い解より良いと推定された `then`
   (b) 評価: 新しい解を評価する
   (c) 更新: 新しい解が古い解より良ければ，古い解を新しい解で置換する
2. `endif`

すなわち，比較推定法では，

- 近似モデルに基づき，新しい解と古い解の良さを推定する．このとき，解の良さは目的関数値を求めずに決める必要がある．
- 新しい解が良いと推定されれば評価・更新を行い，そうでなければ，新しい解は評価することなく拒否する．

ということになる．

3 ポテンシャルを用いた比較推定法 (ポテンシャル法)

3.1 ポテンシャル

ポテンシャルエネルギーとは，物体がある位置に存在することで物体にたくわえられる位置エネルギーである．例えば，次のようなものがある．質量 m の物体が存在すれば，その周りには重力ポテンシャル U_g が発生し，その物体から距離 r 離れた質量 m' の物体との間に万有引力 F_g が生じる．

$$U_g = -G\frac{m}{r},\ F_g = G\frac{mm'}{r^2} \tag{2}$$

ここで，G は万有引力定数である．

本研究では，ある解 $\boldsymbol{x}$ が存在することにより，その解から r 離れた位置に，目的ポテンシャル U_o(potential for objective) と混雑ポテンシャル U_c(potential for congestion) が発生すると仮定する．

$$U_o = \frac{f(\boldsymbol{x})}{r^p},\quad U_c = \frac{1}{r^p} \tag{3}$$

ここでは，簡単のため，比例係数を 1 とした．また，距離のベキ乗数 $p > 0$ は整数値であり通常 1 あるいは 2 とする．解集合 $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N\}$ が存在し，関数値 $f(\boldsymbol{x}_i), i = 1, 2, \cdots, N$ が既知であるとき，ある点 $\boldsymbol{y}$ におけるポテンシャルを，以下のように定義する．

$$U_o(\boldsymbol{y}) = \sum_i \frac{f(\boldsymbol{x}_i)}{d(\boldsymbol{x}_i, \boldsymbol{y})^p},\quad U_c(\boldsymbol{y}) = \sum_i \frac{1}{d(\boldsymbol{x}_i, \boldsymbol{y})^p} \tag{4}$$

ただし，$d(\boldsymbol{x}, \boldsymbol{y})$ は点 $\boldsymbol{x}$ と点 $\boldsymbol{y}$ 間の距離である．

このとき，U_o はある点における関数値の大きさの傾向を示しており，U_c は近傍にどの程度他の点が存在し混雑しているかを示している．点 $\boldsymbol{y}$ における関数の推定値 $\hat{f}(\boldsymbol{y})$ は，これらの商で与えられる．

$$\hat{f}(\boldsymbol{y}) \quad = \quad U_o(\boldsymbol{y})/U_c(\boldsymbol{y}) \tag{5}$$

集団的降下法では，関数値が既知の解集合 X 中の古い解 $\boldsymbol{x}_i$ から新しい解 $\boldsymbol{x}_i'$ を生成する．このとき，それぞれの点における推定値は，以下のように古い解を除いた解集合によるポテンシャルで求めることができる．

$$U_o(\boldsymbol{x}_i^{(\prime)}) = \sum_{j\neq i} \frac{f(\boldsymbol{x}_j)}{d(\boldsymbol{x}_j, \boldsymbol{x}_i^{(\prime)})^p},\ U_c(\boldsymbol{x}_i^{(\prime)}) = \sum_{j\neq i} \frac{1}{d(\boldsymbol{x}_j, \boldsymbol{x}_i^{(\prime)})^p},\ \hat{f}(\boldsymbol{x}_i^{(\prime)}) = U_o(\boldsymbol{x}_i^{(\prime)})/U_c(\boldsymbol{x}_i^{(\prime)}) \tag{6}$$

4 potential DE

Differential Evolution にポテンシャル法を適用したアルゴリズムである potential DE を提案する．

4.1 Differential Evolution

Differential evolution (DE) は進化的戦略 (evolution strategy) の一つであり，Storn and Price[1, 2] によって提案された．DE は確率的な直接探索法であり，解集団を用いた多点探索を行う．DE は非線形問題，微分不可能な問題，非凸問題，多峰性問題などの様々な最適化問題に適用されてきており，これらの問題に対して高速で頑健なアルゴリズムであることが示されてきている．

DE の重要な特徴として，進化的戦略ではガウス突然変異のステップ幅を制御する必要があるが，DE ではこのような制御が不要となる単純な数学的演算を用いていることが挙げられる．一般に，ガウス突然変異における理想的なステップ幅は，遺伝子あるいは各次元毎に異なり，また進化の状態によっても異なるため，何らかの方法でステップ幅を適応的に調整する必要がある．これに対し，DE はガウス突然変異の代わりに，基本ベクトル (base vector) と差分ベクトル (difference vectors) との重み付き和を突然変異として採

用している．集団から選択された1個体が基本ベクトルとなり，集団からランダムに選択された個体対の差が差分ベクトルとなる．世代を経るに従い，解集団が探索空間中で収縮したり拡張したりすることにより，差分ベクトルが変化し，差分ベクトルとして与えられる各次元におけるステップ幅が自動的に調整されるのである．

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．

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となり，残りの個体対が差分ベクトルとなる．差分ベクトルは F(scaling factor) が乗算され基本ベクトルに加えられる．その結果得られたベクトルと親が交叉し，CR(crossover factor) 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．

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．

4.2 potential DE のアルゴリズム

potential DE/rand/1/exp のアルゴリズムは以下のように記述できる [3, 4].

Step0 初期化．N 個の初期個体 $\boldsymbol{x}_i$ を初期探索点として生成し，初期集団 $\{\boldsymbol{x}_i, i = 1, 2, \cdots, N\}$ を構成する．全ての個体を評価する．

Step1 終了判定．終了条件を満足すれば，アルゴリズムは終了する．終了条件としては，最大の繰り返し回数や関数評価回数を用いることが多い．

Step2 突然変異．各個体 $\boldsymbol{x}_i$ に対して，3個体 $\boldsymbol{x}_{p1}$, $\boldsymbol{x}_{p2}$, $\boldsymbol{x}_{p3}$ を $\boldsymbol{x}_i$ および互いに重複しないようにランダムに選択する．新しいベクトル $\boldsymbol{x}'$ を基本ベクトル $\boldsymbol{x}_{p1}$ および差分ベクトル $\boldsymbol{x}_{p2} - \boldsymbol{x}_{p3}$ から以下のように生成する．

$$\boldsymbol{x}' = \boldsymbol{x}_{p1} + F(\boldsymbol{x}_{p2} - \boldsymbol{x}_{p3}) \tag{7}$$

ここで，F はスケーリングパラメータである．

Step3 交叉．ベクトル $\boldsymbol{x}'$ を親 $\boldsymbol{x}_i$ と交叉し，子ベクトル $\boldsymbol{x}_i^{\mathrm{new}}$ を生成する．交差点 j を全ての次元 $[1, n]$ からランダムに選択する．子ベクトル $\boldsymbol{x}_i^{\mathrm{new}}$ の j 番目の要素を $\boldsymbol{x}'$ の j 番目の要素から継承する．それ以降の次元は，交叉パラメータ CR によって指数関数的に減少する確率で，$\boldsymbol{x}'$ の要素から継承する．残りの部分は，親 $\boldsymbol{x}_i$ から継承する．実際の処理では，Step2 と Step3 は一まとまりの処理で実現される．

Step4 ポテンシャル法．もし，子ベクトルが親ベクトルよりも良いと推定されれば，Step5 に進む．そうでなければ，Step6 に進む．

Step5 生存者選択．子ベクトルを評価する．子ベクトル $\boldsymbol{x}_i^{\mathrm{new}}$ が親ベクトルよりも良ければ子ベクトルが生存者となり，親を子ベクトルで置換する．

Step6 Step1 に戻る．

以下に擬似コードを示す.

```
potential DE/rand/1/exp()
{
  P=Generate N individuals {x_i} randomly;
  Evaluate x_i, i = 1,2,...,N;
  for(t=1; 終了条件が満たされない; t++) {
    for(i=1; i ≤ N; i++) {
      (p1,p2,p3)=select randomly from [1,N]\{i}
                s.t. p_j ≠ p_k(j,k = 1,2,3,j ≠ k);
      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});
        j=(j + 1)%n;
        k++;
      } while(k ≤ n && u(0,1) < CR);
      if(Better_Potential(x^new, x)) {
        Evaluate x^new;
        if(f(x_i^new) < f(x_i)) x_i=x_i^new;
      }
    }
  }
}
```

ここで，$u(0,1)$ は区間 $[0,1]$ の一様乱数生成関数，$\mathrm{Better}_{\mathrm{Potential}}(\cdot,\cdot)$ は，ポテンシャル法に基づき，解の良さを比較する関数である．この関数値が常に真ならば，すなわち，$\mathrm{Better}_{\mathrm{Potential}}(\cdot,\cdot)$=1 ならば，常に関数値の評価が行われることになるため，通常の DE/rand/1/exp と一致する．

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 [14]．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

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

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

$$f(\boldsymbol{x}) = \sum_{i=2}^{n} \{100(x_1 - x_i^2)^2 + (x_i - 1)^2\}, \ -2.048 \leq x_i \leq 2.048 \tag{9}$$

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

$$f(\boldsymbol{x}) = \sum_{i=2}^{n} \{100(x_1 - (ix_i)^2)^2 + (ix_i - 1)^2\}, \ -2.048/i \leq x_i \leq 2.048/i \tag{10}$$

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

$$f(\boldsymbol{x}) = 10n + \sum_{i=1}^{n} \{x_i^2 - 10\cos(2\pi x_i)\}, \ -5.12 \leq x_i \leq 5.12 \tag{11}$$

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．

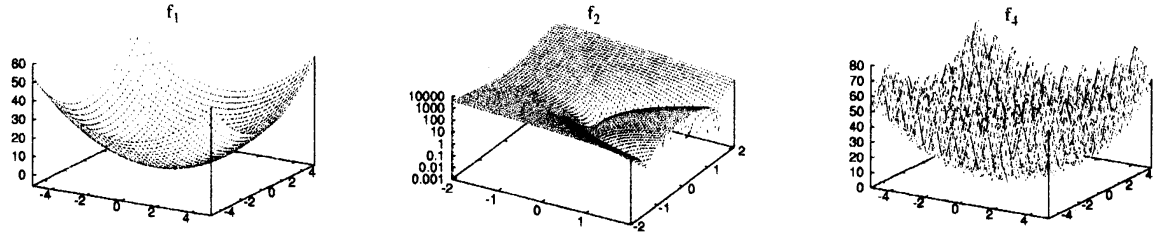

図 1: 関数 f_1, f_2, f_3 のグラフ

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

表 2: DE で良い解を見つける頻度

Func	eval	succ	fail	rate (%)
f_1	76,887.4	10,739.1	66,098.2	13.98
f_2	408,749.4	17,505.7	391,193.8	4.24
f_3	400,122.5	17,292.8	382,779.8	4.32
f_4	275,101.8	14,177.8	260,874.1	5.15

図 1 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．

5.2 予備実験

DE/rand/1/exp を用い，次元数 $n = 30$ に設定し，f_1〜f_4 の関数を最適化する．DE の設定は，個体数 $N = 50$, $F = 0.7$, $CR = 0.95$ とし，各関数について 20 回の試行を行い，結果を考察する．なお，試行打ち切り条件は，文献 [14] 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．

このとき，関数の評価回数を eval に，子ベクトルが親ベクトルより良い解となる成功した関数評価の回数を succ に，失敗した回数を fail に，成功の確率を rate として表 2 に示した．簡単な単峰性の関数でさえ，探索によってより良い解が見つかる確率は 13.98% であり，可能性はあまり高くない．また，より困難な問題では，良い解が見つかる確率は 5% 程度に過ぎない．このように関数評価の多くの部分が悪い解に対するものとなっている．したがって，もし悪い解であることを的確に判断できれば，関数評価回数を大きく削減することが可能となることが期待できる．しかし，一般にこの判断は困難であり，精度の低い判断を行うと，少数の成功した解に対する評価を省略することになり，最適解を見つけられなくなる危険性が高くなる．推定には必ず誤差があるため，誤差を考慮して判断を行う必要があると考えられる．

5.3 potential DE の設定

potential DE の実験条件は，予備実験の時と同様とした．関数 $\mathrm{Better}_{\mathrm{Potential}}$ は推定誤差を考慮して，以下のようにある程度の余裕を与えることにした．

$$\mathrm{Better}_{\mathrm{Potential}}(\boldsymbol{x}_i', \boldsymbol{x}_i) \Leftrightarrow \frac{\hat{f}(\boldsymbol{x}_i') - \hat{f}(\boldsymbol{x}_i)}{|\hat{f}(\boldsymbol{x}_i)|} \leq \delta \tag{12}$$

ただし，$\delta \geq 0$ は誤差の余裕を決めるパラメータであり，推定値を決めるための解集合 X は，各世代における集団 P とした．

δ が 0 の場合は，推定値に基づく比較となり，多数のベクトルを拒否するため，良いベクトルも拒否してしまう可能性が高くなる．δ を大きくすると，推定値が少し悪いベクトルも受け入れるため，良いベクトルを拒否する可能性は低くなるが，逆に，拒否する数が減少する．したがって，δ は適当な大きさにとる必要がある．本実験では，$\delta = 0.001, 0.005, 0.01$ の 3 通りについて調べる．

表 3: 実験結果

Func.	params	eval	succ	fail	rate(%)	obj-succ	obj-fail	obj-rate(%)
f_1	0.001	**33,537.45**	9,852.5	23,634.95	**29.38**	55,491.85	4,061.5	93.18
	0.005	50,058.7	10,600.2	39,408.5	21.18	30,754.25	974	96.93
	0.01	61,135.75	10,642.35	50,443.4	17.41	16,364.4	222.25	**98.66**
f_2	0.001	**352,745**	40,388.9	312,306.2	**11.45**	298,408.2	8,029.1	97.38
	0.005	338,964.4	27,303.35	311,611	8.05	172,743.9	2,040.3	98.83
	0.01	362,287.8	23,994.9	338,242.9	6.62	125,034.1	990.35	**99.21**
f_3	0.001	**339,911.7**	38,727.6	301,134	**11.39**	295,146.6	7,885.05	97.40
	0.005	344,141.3	27,711	316,380.3	8.05	170,904.4	1,979.35	98.86
	0.01	363,576.5	24,233.75	339,292.7	6.67	123,862.9	1,009.65	**99.19**
f_4	0.001	**139,045.2**	13,270	125,725.2	**9.54**	168,580.9	5,006.85	97.12
	0.005	228,197.7	14,236.05	213,911.6	6.24	82,484.15	1,028.45	98.77
	0.01	234,902.3	14,103.3	220,749	7.38	40,538.3	278.05	**99.32**

表 4: 評価回数の比較

Algorithm	Parameters	f_1	f_2	f_3	f_4
DE	potential 0.001	$\mathbf{3.35\times10^4}$	$\mathbf{3.53\times10^5}$	$\mathbf{3.40\times10^5}$	$\mathbf{1.39\times10^5}$
		(0.43)	(0.86)	(0.85)	(0.51)
DE	original	7.69×10^4	4.09×10^5	4.00×10^5	2.75×10^5

さらに，次元毎のスケールの差に対応するために，各世代において集団に存在する解の次元毎の幅，すなわち最大値と最小値の差により正規化した距離を導入する．

$$d(\boldsymbol{x},\boldsymbol{y}) = \sqrt{\sum_j \frac{(x_j - y_j)^2}{(\max_i x_{ij} - \min_i x_{ij})^2}} \tag{13}$$

なお，距離のベキ乗数は，最も計算量が少なくなる $p = 2$ に設定した．

5.4 実験結果

実験結果を表 3 に示す．実際に関数を評価した回数を 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．obj-succ は，拒否されたベクトルが親ベクトルよりも悪かった回数，obj-fail は，拒否されたベクトルが良かった回数，obj-rate はポテンシャル法の成功率，すなわち，拒否したベクトルが実際に悪かった確率を示す．

DE と比較して成功率が倍程度あるいは倍以上に向上し，関数評価回数もかなり減少しており，f_1,f_4 では半分以下に減少している．ポテンシャル法の成功率は 90%以上であり，余裕 (δ) を大きくするにつれて増加するが，関数の成功率は逆に減少する傾向がある．実験では，余裕 $\delta = 0.001$ の場合が最も効率が高い結果となった．

表 4 に potential DE と DE との評価回数の比較を示す．括弧内は DE を 1 としたときの比率である．

potential DE を DE と比較すると，関数評価回数が f_2, f_3 では約 15%削減され，f_1, f_4 では約 50%削減されている．したがって，potential DE は網羅性を損なうことなく，探索効率を大幅に向上させることに成功した，非常に効率性の高いアルゴリズムである．

DE および potential DE($\delta = 0.001$) を比較するために，２つの方法について各問題における関数値の変化および成功率の変化を図 2〜5 の上側，下側にそれぞれ示した．横軸は関数評価回数，縦軸は関数値およ

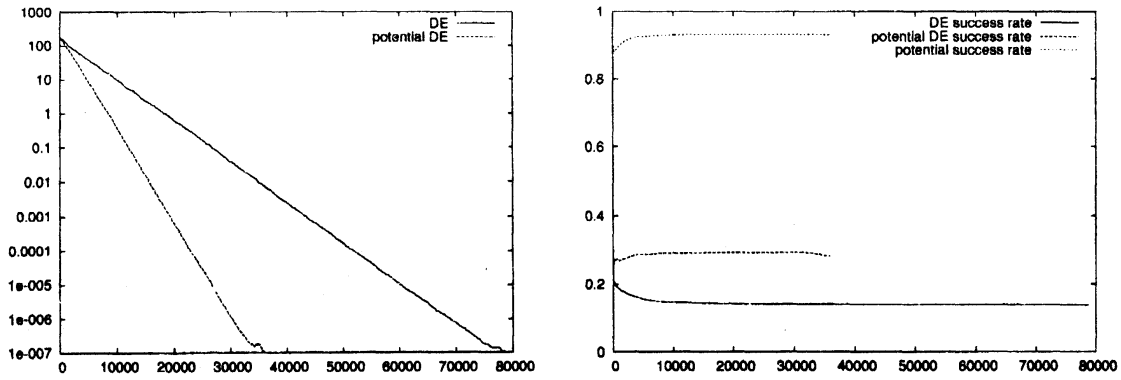

図 2: f_1 における関数値および成功率の変化

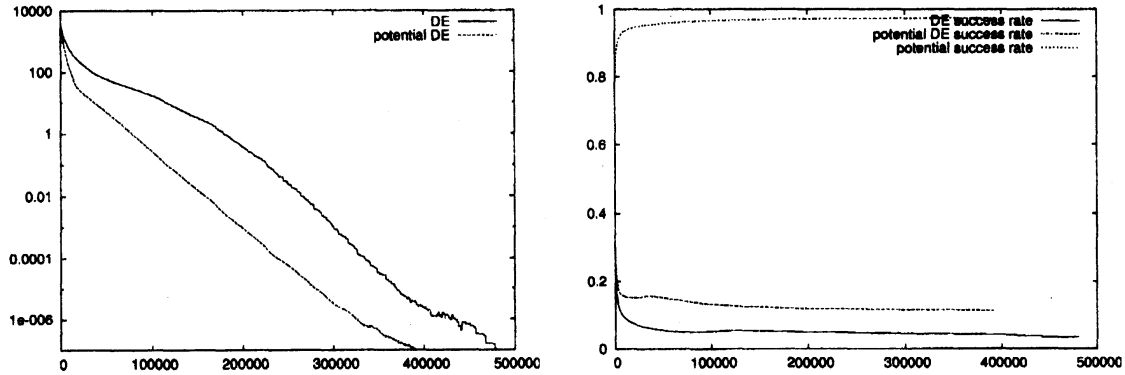

図 3: f_2 における関数値および成功率の変化

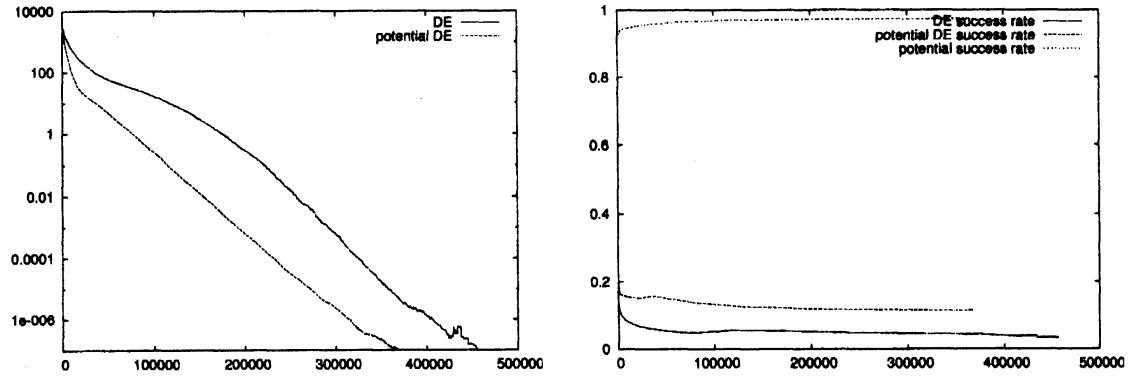

図 4: f_3 における関数値および成功率の変化

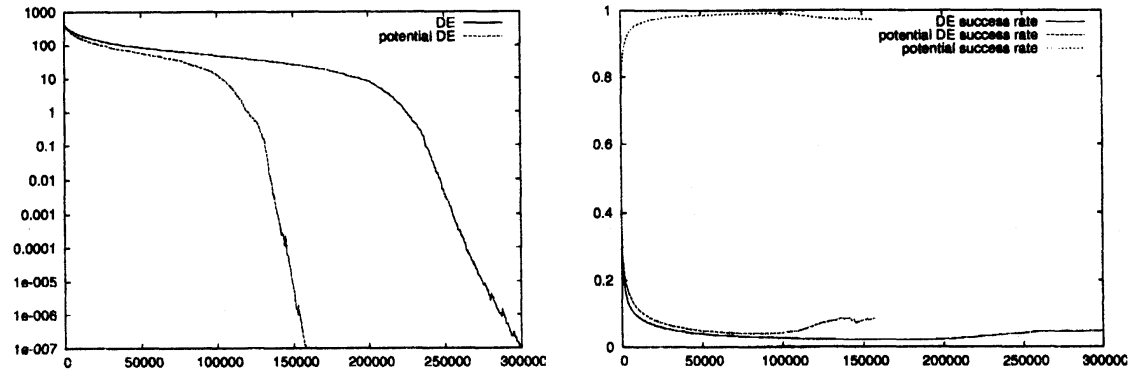

図 5: f_4 における関数値および成功率の変化

び成功率である．potential DE では，ポテンシャル法の成功率を potential success rate に示した．なお，グラフは 20 回の実験の平均値を示しているため，グラフの終末付近では探索を打ち切った試行が多くなり，データ数が減少し平均値に乱れが出ている．

全ての問題において potential DE が DE よりも常に高速に関数値を減少させており，成功率についても，potential DE の方が常に高い値を示している．また，ポテンシャル法の成功率についてもほとんどの場合に 90%を超えており，かなり精度が高いことが分かる．

5.5 議論

ポテンシャルは関数値を滑らかに近似するため，滑らかな関数に対する近似精度は高くなるが，急激に変化する関数に対する近似精度は低下する．実験で用いた関数 f_1 は最も滑らかな関数であり，関数 f_4 も大きな構造としては滑らかなため，解の判定が適切に行われ，評価回数を大きく削減できたと考えられる．

これに対して，f_2 および f_3 は急峻な溝構造を持つため，近似精度が不十分になったと考えられる．実験結果から，$f_2(f_3)$ における DE の成功数が 17,505.7(17,292.8)，potential DE の成功数が 40,388.9(38,727.6) となり，最適解を見つけるまでに倍以上の成功数が必要となっている．これは，ポテンシャル法が最適解に近づくために重要であり本来受け入れるべき解を拒否したため，最適解に近づくのが遅れたのが原因であると考えられる．このように考えると，余裕を大きくとるに従い，必要な成功数が減少しているという現象も説明することができる．

しかし，近似精度が低下した場合に単に余裕を大きくするだけでは必要な成功数は減少するが，評価回数の削減にはつながらないということも実験から明らかである．もちろん，近似精度を向上させるため他の近似方法を採用することも解決策の一つである．しかし，ポテンシャル法において，近似精度が低下した場合に重要な解を拒否しないような方法を考案することは価値があると思われる．

6 おわりに

本研究では，集団的降下法において，関数評価回数を削減し効率性を向上する方法として，ポテンシャル法を提案した．ポテンシャル法は，関数値の近似のために解のポテンシャルを用い，近似値に基づき解を比較し，不必要な解の評価をできる限り行わないようにすることで，関数評価回数を削減する．ポテンシャル法を DE に適用し，従来の DE と比較することにより，potential DE が従来の DE と比較して効率性の高い方法であることを示した．

今後は，ポテンシャル法を PSO などの他の集団的降下法へ適用すること，網羅性についても考慮した比較を導入すること，近似値に基づく比較に Simulated Annealing のような確率的な比較を取り入れることを予定している．

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. 16500083,17510139) の援助のもとで行われた．

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] T. Takahama, S. Sakai and N. Iwane: "Solving nonlinear constrained optimization problems by the ε constrained differential evolution", Proc. of the 2006 IEEE Conference on Systems, Man, and Cybernetics, pp. 2322–2327 (2006).

[4] T. Takahama and S. Sakai: "Constrained optimization by the ε constrained differential evolution with gradient-based mutation and feasible elites", Proc. of the 2006 World Congress on Computational Intelligence, pp. 308–315 (2006).

[5] J. Kennedy and R. C. Eberhart: "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks, Vol. 1498 of Lecture Notes in Computer Science, Perth, Australia, IEEE Press, pp. 1942–1948 (1995). vol.IV.

[6] J. Kennedy and R. C. Eberhart: "Swarm Intelligence", Morgan Kaufmann, San Francisco (2001).

[7] T. Takahama and S. Sakai: "Constrained optimization by combining the α constrained method with particle swarm optimization", Proc. of Joint 2nd International Conference on Soft Computing and Intelligent Systems and 5th International Symposium on Advanced Intelligent Systems (2004).

[8] T. Takahama and S. Sakai: "Constrained optimization by the α constrained particle swarm optimizer", Journal of Advanced Computational Intelligence and Intelligent Informatics, **9**, 3, pp. 282–289 (2005).

[9] T. Takahama and S. Sakai: "Constrained optimization by ε constrained particle swarm optimizer with ε-level control", Proc. of the 4th IEEE International Workshop on Soft Computing as Transdisciplinary Science and Technology (WSTST'05), pp. 1019–1029 (2005).

[10] T. Takahama, S. Sakai and N. Iwane: "Constrained optimization by the ε constrained hybrid algorithm of particle swarm optimization and genetic algorithm", Proc. of the 18th Australian Joint Conference on Artificial Intelligence, pp. 389–400 (2005). Lecture Notes in Computer Science 3809.

[11] T. Takahama and S. Sakai: "Solving constrained optimization problems by the ε constrained particle swarm optimizer with adaptive velocity limit control", Proc. of the 2nd IEEE International Conference on Cybernetics & Intelligent Systems, pp. 683–689 (2006).

[12] Y. Jin: "A comprehensive survey of fitness approximation in evolutionary computation", Soft Computing, **9**, pp. 3–12 (2005).

[13] Y.-S. Ong, Z. Zhou and D. Lim: "Curse and blessing of uncertainty in evolutionary algorithm using approximation", 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, pp. 9833–9840 (2006).

[14] 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).