B-103

ロバスト最適化問題に対する差分進化の適用

Differential Evolution for Robust Optimization Problem

末永 大樹†　　田川 聖治‡
Taiki Suenaga　　Kiyoharu Tagawa

1．　はじめに

様々な現実的な最適化問題では、多様な不確実性要素を考慮する必要がある。不確実性を擁する最適化問題は、a）測定誤差など目的関数に誤差を含む場合、b）加工誤差など決定変数が変動する場合、c）モデル化誤差など目的関数が近似である場合、d）経年変化などにより目的関数が時間と共に遷移する場合に大別される[1]。

このような不確実性を擁する最適化問題に進化計算を適用する際、各個体の目的関数値を複数回のサンプリングにより評価する必要がある。つまり、サンプリング回数によって得られる解の精度は左右される。しかし、サンプリングによる解の評価は計算コストが伴う。そのため、不確実性を擁する最適化問題に対して効率的な進化計算を構築する上で、解の精度を損なわずサンプリング回数を減らす工夫が重要となる。そこで、様々なサンプリング回数の削減方法が提案されている[1]。例えば、喜多らは、上記の a）の場合の目的関数に誤差を含む最適化問題を対象に、個体間の距離に基づく確率モデルと探索履歴による目的関数値の推定法を提案している[2]。

差分進化（DE：Differential Evolution）[3]とは決定変数が実数値を取る関数最適化問題に対する進化計算の一種である。DE はアルゴリズムが単純であるため実装が容易である。また、代表的なテスト関数を対象とした数値実験によれば、典型的な遺伝的アルゴリズムや進化戦略と比較し、DE は解の収束に優れ、得られる解も頑健ある[4]。DE の遺伝子的演算子である戦略では、ターゲットベクトルと呼ばれる 1 つの親個体よりトライアルベクトルと呼ばれる 1 つの子個体が生成される。次に、DE の生存選択では、ターゲットベクトルとトライアルベクトルを比較することで、親子間でのトーナメント選択を行う。

本研究では、b）の場合の決定変数が変動する最適化問題を対象に、新たな DE のアルゴリズムを提案する。ここで、b）の問題はロバスト最適化問題とも呼ばれる。

本稿で提案する DER（DE for Robust optimization）では、生存選択に着目し、トライアルベクトルに対する目的関数値を予測区間[6]により評価し、サンプリング回数を削減する。最後に、従来の DE と比較して、DER では解の精度を損なわず、高速に解が得られることを確認する。

2．ロバスト最適化問題

ロバスト最適化問題の目的関数 $F(\vec{x})$ は各決定変数 x_j がノイズ $\delta_{j,t}$ を含み、式(1)のように記述される。

$$F(\vec{x}) = F(x_0, \cdots, x_j, \cdots, x_{D-1}) = f(\vec{x}+\vec{\delta}_t)$$
$$\vec{\delta}_t = (\delta_{0,t}, \cdots \delta_{j,t} \cdots, \delta_{D-1,t}) \quad (1)$$

ある解に対して目的関数値を N 回評価して標本とすると、それらの標本平均は式(2)のように求められる。

$$\overline{f}(\vec{x}) = \frac{1}{N}\sum_{t=1}^{N} f(\vec{x}+\vec{\delta}_t) \quad (2)$$

ロバスト最適化問題に対する既存のアルゴリズムでは、式(2)の標本平均を目的関数として、その値を最小とする解 $\vec{x}$ を探索していた[1]。しかし、標本平均は解の標本の平均的な良さであり、標本のバラツキや目的関数値の最悪値を考慮することができない。ちなみに、標本のバラツキは式(3)の標本不変分散によって評価できる。

$$s(\vec{x}) = \sqrt{\frac{\sum_{t=1}^{N}(f(\vec{x}+\vec{\delta}_t) - \overline{f}(\vec{x}))^2}{N-1}} \quad (3)$$

ある解 $\vec{x}$ に対して N 個の目的関数値 $f(\vec{x}+\vec{\delta}_t)$ が標本として得られているとき、その解 $\vec{x}$ に対する目的関数値の $t=N+1$ 番目の標本の値 $f(\vec{x}+\vec{\delta}_{N+1})$ は、有意水準を α として式(4)の範囲にあると予測される[5]。

$$\overline{f}(\vec{x}) - \beta\, s(\vec{x}) < f(\vec{x}+\vec{\delta}_{N+1}) < \overline{f}(\vec{x}) + \beta\, s(\vec{x})$$
$$\beta = t(N-1;\alpha)\sqrt{1+\frac{1}{N}} \quad (4)$$

ただし、$t(N-1;\alpha)$ は有意水準を α とする t 分布の上側確率である。また、本稿では $a = 0.05$ とする。

本稿では、式(4)の予測区間の上限値を目的関数とする以下のロバスト最適化問題を対象とする。

$$\text{Minimize } F(\vec{x}, N) = \overline{f}(\vec{x}) + \beta\, s(\vec{x})$$
$$\text{Subject } \underline{x}_j \le x_j \le \overline{x}_j \quad (5)$$

上記のロバスト最適化問題において、決定変数は D 個の実数値 $x_j \in \Re \ (j = 0, \cdots, D-1)$ であり、それらの上下限値が $\underline{x}_j$ と $\overline{x}_j$ である。また、$\delta_{j,t}$ は標準正規分布 $\mathrm{N}(0,1)$ に従って観測毎に独立な値を取るものとする。

式(5)のロバスト最適化問題では、誤差を含む目的関数値の平均的な良さである標本平均と、そのバラツキの評価指標である標本不変分散を総合的に考慮している。

†近畿大学総合理工学研究科，Graduate School of Science and Engineering Research, Kinki University

‡ 近畿大学理工学部，School of Scienc and Engineering, Kinki University

3．Differential Evolution (DE)

進化計算に対する個体集団の更新方法は世代交代モデルと呼ばれ、離散世代モデル（Generational model）と連続世代モデル（Steady-state model）に大別される[6]。DE は、初期では離散世代モデルを採用していたが[3]、その後、連続世代モデルに基づく DE も報告されている[7][8]。

本稿では、連続世代モデルに基づく DE を採用するものとして、以下に式(5)で定義したロバスト最適化問題に対して DE をそのまま適用するアルゴリズムを説明する。

3.1 個体表現と集団構造

DE では、式(5)のロバスト最適化問題に対する解候補を個体とし、それらの個体の配列を集団 P とする。すなわち、集団の個体数を N_p とすると、集団内の i 番目の個体 $\vec{x}_i \in P\,(i=0,\cdots,N_p-1)$ は、浮動小数点表現による実数ベクトルであり、式(6)にように表される。

$$\vec{x}_i = (x_{0,i},\cdots x_{j,i,}\cdots,x_{D-1,i}) \tag{6}$$

ただし、$\underline{x}_j \le x_{j,i} \le \bar{x}_j\ (j=0,\cdots,D-1)$ とする。

3.2 DE の戦略

DE では、集団の個体 $\vec{x}_i\ (i=0,\cdots,N_{p-1})$ を順番にターゲットベクトル $\vec{x}_i$ に指定し、それに戦略を適用することで、新たな個体候補となるトライアルベクトル $\vec{u}$ を生成する。DE の戦略は幾つか報告されているが[4]、本稿では「DE/rand/1/exp」を採用し、図 1 に擬似コードを示す。

```
j=jr;
do{
  u_j = x_{j,r1} + S_F(x_{j,r2} - x_{j,r3});
  j = (j+1)%D;
}while(rand_j[0,1] < C_R ∧ j ≠ j_r)
while(j ≠ j_r){
  u_j = x_{j,i};
  j = (j+1)%D;
}
```

図 1　「DE/rand/1/exp」の擬似コード

図 1 に示した擬似コードにおいて、$\mathrm{rand}_j[0,1]$ は範囲 $[0,1]$ の一様乱数である。また、記号「%」はモジュロ演算である。集団内からランダムに選択した 3 つの異なる個体を $\vec{x}_{r1},\vec{x}_{r2},\vec{x}_{r3}\ (i\neq r1\neq r2\neq r3)$ とする。スケール係数 $S_F(S_F\ge 0)$ と交叉率 $C_R(0\le C_R\le 1)$ は、ユーザが事前に与える DE の制御パラメータである。通常、スケール係数 S_F は定数であるが、集団の収束とともに個体間のベクトル差が小さくなるため、$\vec{x}_{r1}$ に加わる摂動の大きさが適切に調節されることが期待できる。変数の添え字 $j_r(0\le j_r<D)$ はランダムに選択される。このため、トライアルベクトルは少なくとも 1 つの要素において、ターゲットベクトルと異なることが保証される。

図(1)の戦略により、$\vec{u}$ の要素 u_j が探索範囲 $[\underline{x}_j,\bar{x}_j]$ から外れた場合は、以下の式(7)より u_j を修正する。

$$u_j = (\bar{x}_j - \underline{x}_j)\ rand_j\,[0,1] + \underline{x}_j \tag{7}$$

3.3 DE の処理手順

式(5)のロバスト最適化問題に対する DE の擬似コードを図 2 に示す。はじめに、初期集団の個体をランダムに生成し、各個体に対して式(5)で定義した目的関数を計算する。次に、図 1 に示した戦略によりトライアルベクトル $\vec{u}$ を生成し、その目的関数値を計算する。DE の生存選択ではターゲットベクトル $\vec{x}_i$ とトライアルベクトル $\vec{u}$ の目的関数値を直接比較する親子間のトーナメント選択を行う。DE の終了条件として世代数の最大値 G_M を指定する。このため、世代数が最大値に達すると、DE は集団内で目的関数値が最小の個体を出力して終了する。

```
for (i=0; i<N_p; i++){
  Randomly generate x_i ∈ P;
  Evaluate F(x_i, N);
}
for(g=0; g<G_M; g++){
  for(i=0; i<N_p; i++){
    Generate u;
    Evaluate F(u, N);
    if(F(u, N) ≤ (F(x_i, N))){
      x_i = u;
      F(x_i, N) = F(u, N);
    }
  }
}
Output the best x_i ∈ P;
```

図 2　DE の疑似コード

4．DE for Robust Optimization (DER)

ロバスト最適化問題において解の精度を高めるには、式(2)のサンプリング回数 N を多くする必要がある。しかし、図 2 に示した DE では、各個体の評価において、式(1)の関数値を N 回計算する必要があり、サンプリング回数 N を増やすと DE の実行時間が膨大となる。

本稿では、ロバスト最適化問題に対する DE の効率的な構築法として、新たに DER を提案する。DER の擬似コードを図 3 に示す。まず、3.3 節と同様に、初期集団の個体を生成し、各個体の目的関数を計算する。次に、トライアルベクトル $\vec{u}$ を生成し、図 3 に示すように、式(1)の $F(\vec{u})$ を計算後、$F(\vec{u})$ と $\vec{x}_i$ の $F(\vec{x},N)$ と比較する。$F(\vec{u})$ より、$F(\vec{x}_i,N)$ が劣れば計算を省略し、$F(\vec{x}_i,N)$ が勝れば 3.3 節と同様に DE の生存選択の処理に戻り、世代数が最大値に達すると、集団内で目的関数値が最小の個体を出力して終了する。つまり、生成されたトライアルベクトルが予測区間による上限値未満であ

れば、以降の個体群の計算手順を省略し計算コストを減少させる仕組みである。

```
for (i = 0; i < N_p; i++) {
    Randomly generate x_i ∈ P;
    Evaluate F(x_i, N);
}
for (g = 0; g < G_M; g++) {
    for (i = 0; i < N_p; i++) {
        Generate u;
        Evaluate F(u);
        if (F(u) < (F(x_i, N)) {
            Evaluate F(u, N);
            if (F(u, N) ≤ (F(x_i, N)) {
                x_i = u;
                F(x_i, N) = F(u, N);
            }
        }
    }
}
Output the best x_i ∈ P;
```

図 3　DER の疑似コード

5. 実験と考察

3 章と 4 章で示した疑似コードに基づき DE と DER のプログラムを Java 言語で実装した。プログラムの実行に用いたパソコンの CPU は Intel®Xeon®,E5-2660@2.2[GHz]であり、メモリは 32GB、基本ソフトウェアは Windows 7 である。

5.1　テスト関数

ロバスト最適化問題の目的関数 $F(\vec{x},N)$ において $f(\vec{x})$ を定義するため、以下の 6 種類のテスト関数 $f_p(\vec{x})$ を用意した。なお、サンプル数は $N=100$ とし、テスト関数の次元はすべて $D=20$ とする。

- Sphere 関数：単峰性

$$f_1(x)=\sum_{j=0}^{D-1}x_j^2$$

$$-100\le x_j\le 100,\ j=0,\cdots,D-1$$

- Salomon 関数：多峰性

$$f_2(x)=1+0.1\sqrt{\sum_{j=0}^{D-1}x_j^2}-\cos\left(2\pi\sqrt{\sum_{j=0}^{D-1}x_j^2}\right)$$

$$-100\le x_j\le 100,\ j=0,\cdots,D-1$$

- Rosenbrock 関数：変数間依存性

$$f_3(x)=\sum_{j=0}^{D-1}(100(x_0-x_j^2)^2+(1-x_j)^2)$$

$$-100\le x_j\le 100,\ j=0,\cdots,D-1$$

- Rastrigin 関数：多峰性

$$f_4(x)=\sum_{j=0}^{D-1}(x_j^2-10\cos(2\pi x_j)+10)$$

$$-5.12\le x_j\le 5.12,\ j=0,\cdots,D-1$$

- Ackley 関数：多峰性

$$f_5(x)=20-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{j=0}^{D-1}x_j^2}\right)+e-\exp\left(\frac{1}{D}\sum_{j=0}^{D-1}\cos(2\pi x_j)\right)$$

$$-32.768\le x_j\le 32.768,\ j=0,\cdots,D-1$$

- Ackley 関数：多峰性

$$f_6(x)=\frac{1}{4000}\sum_{j=0}^{D-1}x_j^2-\prod_{j=0}^{D-1}\cos\left(\frac{x_j}{\sqrt{j+1}}\right)+1$$

$$-600\le x_j\le 600,\ j=0,\cdots,D-1$$

5.2　数値実験

DE と DER ともに、制御パラメータは集団の個体数 $N_p=100$、スケール係数 $S_F=0.5$、交叉率 $C_R=0.9$、世代数 $G_M=1000$ とする。ここで、DE と DER を上記 6 種類のテスト関数 $f_P(\vec{x})\,(p=1,\cdots,6)$ に基づき誤差を付与したロバスト最適化問題に対して 30 回ずつ適用した。表 1 に、両者の実行時間の平均を示す。同様に最良解 $\vec{x}^*$ に対する目的関数値 $F_p(\vec{x}^*,N)$ の平均値を表 2 に示す。

表 1 より、テスト関数 f_2 を除く全てのテスト関数において、既存の DE より提案した DER では実行時間が短縮されることが確認できる。また、表 2 の数値実験の結果から、全てのテスト関数において、DE と DER により得られた解の質に大きな差はないように思われる。

5.3　統計的検定

数値実験の結果を統計的検定によって検証する。母集団に正規性が仮定できないため、ノンパラメトリック検定の 1 種であるウィルコクソン（Wilcoxon）の順位和検定

[9]を使用する。表 2 の目的関数値の比較に順位和測定を適用した結果が表 3 である。表 3 の左反面に DE と DER の平均順位和を示している。また、表 3 の右反面に判定結果として、DER または DE が危険率 5%で他方に勝っていることを「*」印で示している。

表 3 から、テスト関数 f_5 において危険率 5%で、DER より DE によって求めた解の精度が高いことがわかる。しかし、他のテスト関数では DE と DER による解の精度に大きな差は見られず、提案した DER は、解の精度を落とさず実行速度を高速化できるといえる。

表 1 ：実行時間の比較[ms]

f_p	DE	DER
f_1	6663.967	4935.733
f_2	5140.533	5329.400
f_3	5054.300	3702.567
f_4	8128.300	2661.067
f_5	7796.300	5083.100
f_6	8069.633	7931.767

表 2：目的関数値の比較

f_p	DE	DER
f_1	0.248	0.250
f_2	2.166	2.163
f_3	43.093	42.986
f_4	71.802	72.760
f_5	0.879	0.886
f_6	8069.633	7931.767

表 3：目的関数値の順位和検定

f_p	DE	DER	DE	DER
f_1	26.63	34.37		
f_2	31.43	29.57		
f_3	31.60	29.40		
f_4	28.03	32.97		
f_5	25.60	35.40	*	
f_6	30.63	30.37		

5.4 考察

最後に、テスト関数 f_2 における、DER による実行速度の低下について考察する。式(1)で、付与するノイズは正規分布に従い、式(5)の目的関数の生成において、式(1)を利用している。つまり、ノイズを含む変数を引数として目的関数を生成しているため、テスト関数 f_2 のような複雑な関数を用いた場合、目的関数の値が必ずしも正規分布に従うとは限らない。次に、式(4)の予測区間は、標本が正規分布に従うことを仮定して導出している。そのため、目的関数値の値が正規分布と大きく異なる場合、予測区間は意味をなさず、式(1)の値の計算に加えて式(5)の目的関数値を計算するトライアルベクトルが増える。そのため、テスト関数 f_2 では計算コストが増加したと考えられる。

6. おわりに

本稿では、ロバスト最適化問題に適用する DE の実行時間を短縮するため、新たに DE を拡張した DER を提案した。数値実験と統計的検定の結果より、提案した DER は実行時間の短縮に効果が見られることを確認した。

考察で述べたように、目的関数の計算において、ノイズを含む変数を引数として使用しているので、必ずしも正規分布に従う目的関数値が得られるとは限らない。今後の課題として、予測区間をノンパラメトリックな方法で算出するなどの対策が必要である。また、実際のロバスト最適化問題[10]で DER を評価する必要もある。

参考文献

[1] Y. Jin and J. Branke：“Evolutionary optimization in uncertain environments－a survey,” IEEE Trans. on Evolutionary Computation, Vol. 9, No. 3, pp. 303- 317 (2005)

[2] 喜多一・佐野泰仁：「不確実環境下での遺伝的アルゴリズム；応用の視点から」, 電気学会論文誌C, 121巻, 6号, pp. 982- 985 (2001)

[3] R. Storn and K. Price：“Differential evolution - a simple and efficient heuristic for global optimization over continuous space,” Journal of Global Optimization, Vol. 11, No. 4, pp. 341- 359 (1997)

[4] K. V. Price, R.M. Storn, and J. A. Lampinen：DifferentialEvolution - A Practical Approach to Global Optimization, Springer (2005)

[5] Wikipedia: “Prediction interval” (http://en.wikipedia.org./wiki/Prediction_interval) (2013)

[6] G. Syswerda：“A study of reproduction in generational and steady-state genetic algorithms,” *Foundations of Genetic Algorithms 2*, Morgan

[7] V. Feoktistov：Differential Evolution in Search Solutions, Springer (2006)

[8] K. Tagawa：“A statistical study of the differential evolution based on continuous generation model,” Proc. of IEEE Congress on Evolutionary Computation, pp. 2614- 2621 (2009)

[9] 喜田安哲：データ分析とSPSS2；展開編, 北樹出版 (2006)

[10] 田川聖治・大谷透・井垣努・関俊一・井上克己：「ペナルティ関数法によるSAWフィルタのロバスト最適設計」, 電気学会論文誌C, 126 巻, 1 号, pp. 1- 7 (2006)