集団的降下法による最適化
—差分進化と粒子群最適化—

高濱徹行 (広島市立大学)

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，以下の 2 点が重要とされている．

- 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，差分進化 (Differential Evolution, DE) と 粒子群最適化 (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．

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．

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

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，$g_j(\boldsymbol{x}) \le 0$ 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．ここでは，不等式制約および等式制約のない問題の最適化について説明する．

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 (population-based descent method) 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

3 差分進化

3.1 特徴

差分進化 (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．

3.2 定義

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 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．

3.3 アルゴリズム

DE/rand/1/{bin,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．

$$\boldsymbol{x}' = \boldsymbol{x}_{p1} + F(\boldsymbol{x}_{p2} - \boldsymbol{x}_{p3}) \tag{2}$$

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．交叉法が bin の場合には，それ以降の次元は，交叉率 CR の確率で，$\boldsymbol{x}'$ の要素から継承し，$1-CR$ の確率で親 $\boldsymbol{x}_i$ から継承する．交叉法が exp の場合には，それ以降の次元は，CR によって指数関数的に減少する確率で，$\boldsymbol{x}'$ の要素から継承し，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．

```
DE/rand/1/bin()
{
  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++) {
      (p_1,p_2,p_3)=select randomly from [1,N]
                s.t. p_1 ≠ p_2 ≠ p_3 ≠ i;
      j=select randomly from [1,n];
      for(k=1; k ≤ n; k++) {
        if(k == 1 || u(0,1)<CR)
          x_j^new=x_{p_1,j}+F(x_{p_2,j} − x_{p_3,j});
        else
          x_j^new=x_ij;
        j=(j + 1)%n;
      }
      Evaluate x^new;
      if(f(x^new) < f(x_i)) x_i=x^new;
      }
    }
  }
}

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++) {
      (p_1,p_2,p_3)=select randomly from [1,N]\{i}
                s.t. p_j ≠ p_k(j,k = 1,2,3, j ≠ k);
      j=select randomly from [1,n];
      k=1;
      do {
        x_j^new=x_{p_1,j}+F(x_{p_2,j} − x_{p_3,j});
        j=(j + 1)%n;
        k++;
      } while(k ≤ n && u(0,1) < CR);
      for(; k ≤ n; k++) {
        x_j^new=x_ij;
        j=(j + 1)%n;
      }
      Evaluate x^new;
      if(f(x^new) < f(x_i)) x_i=x^new;
```

```
      }
    }
  }
}
```

ここで，$u(0,1)$ は区間 $[0,1]$ の一様乱数生成関数である．

4 粒子群最適化 (Particle Swarm Optimization)

粒子群最適化 (Particle Swarm Optimization, PSO) による最適化について説明する．

4.1 定義

エージェントのグループがある目的関数 f を最適化すると仮定する．各エージェント i は，時刻 t における各自の位置 $\boldsymbol{x}_i^t$，移動速度 $\boldsymbol{v}_i^t$，および今まで経験した目的関数の最良値 $pbest_i$ とそのときの位置 $\boldsymbol{x}_i^*$ を記憶している．

$$pbest_i = \min_{\tau=0,1,\cdots,t} f(\boldsymbol{x}_i^\tau) \tag{3}$$

$$\boldsymbol{x}_i^* = \arg\min_{\tau=0,1,\cdots,t} f(\boldsymbol{x}_i^\tau) \tag{4}$$

さらに，各エージェントは，グループ中のエージェントが経験した目的関数の最良値 $gbest$ とそのときの位置 $\boldsymbol{x}_G^*$ の情報を共有する．

$$gbest = \min_i pbest_i \tag{5}$$

$$\boldsymbol{x}_G^* = \arg\min_i f(\boldsymbol{x}_i^*) \tag{6}$$

このとき，時刻 $t+1$ におけるエージェントの移動速度は，以下のように求められる．

$$\begin{aligned} v_{ij}^{t+1} = w v_{ij}^t &+ c_1\, rand\,(x_{ij}^* - x_{ij}^t) \\ &+ c_2\, rand\,(x_{Gj}^* - x_{ij}^t) \end{aligned} \tag{7}$$

ただし，w は慣性重み (inertia weight)，$rand$ は区間 $[0,1]$ の一様乱数であり，各成分毎に生成する．c_1 は “cognitive”，c_2 は “social” とよばれるパラメータであり，自己の最良位置およびグループの最良位置への探索に対する重み付けを表現している．

式 (7) から，時刻 $t+1$ におけるエージェントの位置が以下のように求められる．

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{8}$$

4.2 アルゴリズム

PSO のアルゴリズムを以下に示す．

1. エージェントの初期化：位置 $\boldsymbol{x}_i$ と移動速度 $\boldsymbol{v}_i$ を有するエージェント i を生成し，経験した最良位置 $\boldsymbol{x}_i^*$ の初期値を $\boldsymbol{x}_i$ とする．ただし，$\boldsymbol{x}_i$ は上下限制約領域 S 内にランダムに生成する，すなわち，各要素 x_{ik} を区間 $[l_k, u_k]$ の一様乱数とする．$\boldsymbol{v}_i$ の各要素 v_{ik} は，0 とする．
2. 最良エージェントの決定：最良のエージェント G を決定する．
3. 終了判定：本論文では最大反復回数 T に達したとき，実行を終了する．
4. エージェントの更新：各エージェント i について，式 (7),(8) により移動速度および位置を更新する．なお，速度が大きくなりすぎないように，各次元の速度を区間 $[-V_{max_j}, V_{max_j}]$ に入るように調整する．新しい位置における目的関数値が経験した最良値よりも良ければ，新しい位置を最良位置とする．さらに，新しい位置がグループの最良位置よりも良ければ，その位置をグループ最良位置とする．
5. 3. へ戻る．

PSO のアルゴリズムを以下に C 言語風に記述する．

```
PSO()
{
  Initialize Agents(0);
  Evaluate all x in Agents(0);
  x*_G=arg min f(x), x in Agents(0);
  for(t=1;t≤T;t++) {
    w=w^0+(w^T−w^0)(t−1)/(T−1);
    for(each agent i in Agents(t)) {
      for(each dimension j) {
        v_ij=wv_ij+c_1u(0,1)(x*_ij−x_ij)+c_2u(0,1)(x*_Gj−x_ij);
        if(v_ij>V_j) v_ij=V_j;
        else if(v_ij<−V_j) v_ij=−V_j;
        x_ij=x_ij+v_ij;
      }
      Evaluate x_i;
      if(f(x_i)<f(x*_i)) {
        if(f(x_i)<f(x*_G)) G=i;
        x*_i=x_i;
      }
    }
  }
}
```

References

[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).