2002年日本オペレーションズ・リサーチ学会
秋季研究発表会

ガウス型関数の割り当てによる多峰性関数の最適化

01204675　茨城大学　＊佐藤 泰司　SATOH Taiji
01403650　茨城大学　　奈良 宏一　NARA Koichi
01606370　茨城大学　　三島 裕樹　MISHIMA Yuji

1　はじめに

連続型最適化問題の大域的探索手法としてトンネリングアルゴリズム [1] が提案されている。これにヒントを得て、得られた局所解に極を配置し、局所探索手法と組み合せることにより、複数の局所解を探索する手法 [2] が提案されている。しかし、この手法では極の近傍でペナルティ関数が非常に大きくなってしまうため、数値的に不安定となる。そこで、本研究では、連続型最適化問題に対して、局所解に対する極の配置ではなくガウス型関数の配置により解く方法を提案する。

2　ガウス関数

一般に、n 次元のガウス関数はベクトル表示を用いれば、次式のように表わされる。

$$p(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n |\Lambda|}} \exp\left\{ -\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T \Lambda^{-1} (\boldsymbol{x}-\boldsymbol{\mu})}{2} \right\} \quad (1)$$

ここで、$\boldsymbol{x}$ は x_i を要素とするベクトルであり、$\boldsymbol{\mu}$ は $\boldsymbol{x}$ の平均値 μ_i を要素とするベクトルである。また、Λ は共分散行列である。

3　ガウス型関数の配置

(1) 式は確率密度関数としてのガウス関数であるが、以下では、最適化問題に対する応用のみを考えるため、ガウス関数が確率密度関数であるための正規条件を取り除いて議論を進める。そのため、以下ではそのような関数をガウス型関数と呼ぶことにする。つまり、n 変数のガウス型関数を

$$p(\boldsymbol{x}) = K \exp\left\{ -\frac{(\boldsymbol{x}-\boldsymbol{\mu})^T \Gamma (\boldsymbol{x}-\boldsymbol{\mu})}{2} \right\} \quad (2)$$

で定義する。このとき、関数 $p(\boldsymbol{x})$ は $\boldsymbol{x} = \boldsymbol{\mu}$ で、最大値 $p(\boldsymbol{\mu}) = K$ を持つ。また、Γ はガウス型関数のすそ野の広がり具合を表す正定値行列であり、(1) 式の Λ^{-1} に対応する。

局所探索法により連続型最適化問題

$$\underset{\boldsymbol{x} \in X}{minimize}\ f(\boldsymbol{x}) \quad (3)$$

の局所解 $\boldsymbol{x}^*$ が得られたものとする。ここで、X は問題の許容領域である。このとき、次のペナルティ関数を考える。ここで、$p(\boldsymbol{x})$ は局所解 $\boldsymbol{x}^*$ に対応する谷を埋めるためのガウス型関数であり、(2) 式を用いて

$$p(\boldsymbol{x}) = K \exp\left\{ -\frac{(\boldsymbol{x}-\boldsymbol{x}^*)^T \Gamma (\boldsymbol{x}-\boldsymbol{x}^*)}{2} \right\} \quad (4)$$

で定義される。ここで、Γ と K を変化させた場合の例を各々図 1 および図 2 に示す。

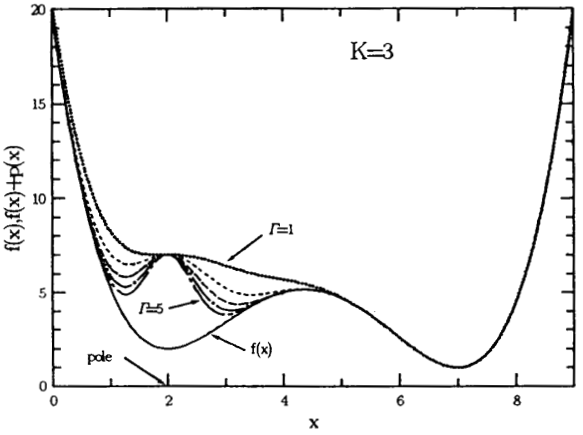

図 1: ガウス型関数の割り当て（Γ の変化）

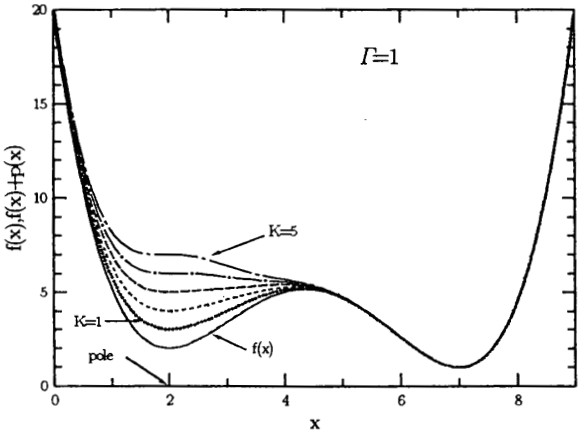

図 2: ガウス型関数の割り当て（K の変化）

3.1　アルゴリズム

ガウス型関数を利用したアルゴリズムのプロトタイプを以下に示す。

適応的ガウス型関数配置アルゴリズム

Step 0: 初期化として、局所解はまだ見つかっていないものとし、局所解のリストを空とする。

Step 1: 探索領域内のランダムな初期値から (5) 式のペナルティ付きの目的関数に対して局所探索法を

適用する。なお、以下の手続きで Step 5 が規定回数連続して失敗すれば、手続きを終了する。

$$\underset{\boldsymbol{x} \in X}{minimize}\ f(\boldsymbol{x}) + p(\boldsymbol{x}) \tag{5}$$

Step 2: Step 1 の局所解から、(3) 式の目的関数に対して局所探索法を適用する。

Step 3: Step 2 で得られた解が、それまでに求まっている局所解と一致するか否かを調べる。

Step 4: 一致しなければ、Step 2 で得られた解は新しい局所解であるので、局所解のリストに加え、Step 1 に戻る。そうでなければ、Step 5 へ。

Step 5: 今までに求まっている局所解と一致すれば、その局所解に対する強度が不足しているので、局所解の強度を 2 倍にし、Step 1 に戻る。

3.2 数値計算

このアルゴリズムの実行結果を以下に示す Camel 関数、Himmelblau 関数、Kearfott 関数 [3] に適用した結果、Camel 関数では 6 個の局所解がすべて求まった。また、Himmelblau 関数および Kearfott 関数では 4 個の最適解がすべて求まった。ただし、ここでは初期値として $K = 1$ とし、$\Gamma = E$ は不変とした。

例 3.1 (Six-Hump Camel-Back Function) この場合の $f(\boldsymbol{x})$ は

$$f(\boldsymbol{x}) = [4 - 2.1x_1^2 + (x_1^4/3)]\, x_1^2 + x_1 x_2 + (-4 + 4x_2^2)\, x_2^2$$

この関数は 6 個の局所最適解を持ち、プログラムの実行結果を以下に示す。

```
New Solution Found.
  0: -1.6071E+00 -5.6865E-01
New Solution Found.
  0: -1.6071E+00 -5.6865E-01
  1: -1.7036E+00  7.9608E-01
New Solution Found.
  0: -1.6071E+00 -5.6865E-01
  1: -1.7036E+00  7.9608E-01
  2:  1.7036E+00 -7.9608E-01
New Solution Found.
  0: -1.6071E+00 -5.6865E-01
  1: -1.7036E+00  7.9608E-01
  2:  1.7036E+00 -7.9608E-01
  3: -8.9842E-02  7.1266E-01
New Solution Found.
  0: -1.6071E+00 -5.6865E-01
  1: -1.7036E+00  7.9608E-01
  2:  1.7036E+00 -7.9608E-01
  3: -8.9842E-02  7.1266E-01
  4:  8.9842E-02 -7.1266E-01
Drop into Known Local Minimum : 2.
Drop into Known Local Minimum : 1.
Drop into Known Local Minimum : 4.
Drop into Known Local Minimum : 1.
Drop into Known Local Minimum : 1.
Drop into Known Local Minimum : 3.
Drop into Known Local Minimum : 4.
Drop into Known Local Minimum : 0.
Drop into Known Local Minimum : 3.
New Solution Found.
  0: -1.6071E+00 -5.6865E-01
  1: -1.7036E+00  7.9608E-01
  2:  1.7036E+00 -7.9608E-01
  3: -8.9842E-02  7.1266E-01
  4:  8.9842E-02 -7.1266E-01
  5:  1.6071E+00  5.6865E-01
Drop into Known Local Minimum : 0.
Drop into Known Local Minimum : 3.
Drop into Known Local Minimum : 5.
Drop into Known Local Minimum : 2.
Drop into Known Local Minimum : 1.
Drop into Known Local Minimum : 5.
Drop into Known Local Minimum : 5.
Drop into Known Local Minimum : 4.
Drop into Known Local Minimum : 5.
Drop into Known Local Minimum : 0.
Iteration Over.
```

このうちの $\boldsymbol{x}^3$ と $\boldsymbol{x}^4$ の 2 個が大域的最適解である。

例 3.2 (Himmelblau 関数) この場合、$f(\boldsymbol{x})$ は

$$f(\boldsymbol{x}) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$$

で与えられる。この関数は 4 個の大域的最適解を持つ。

$$\boldsymbol{x}_1 = (3, 2)$$
$$\boldsymbol{x}_2 = (3.584428340330, -1.848126526964)$$
$$\boldsymbol{x}_3 = (-3.779310253378, -3.283185991286)$$
$$\boldsymbol{x}_4 = (-2.805118086953, 3.131312518250)$$

例 3.3 (Kearfott 関数) 目的関数 $f(\boldsymbol{x})$ は

$$f(\boldsymbol{x}) = \left(x_1^2 + x_2^2 - 2\right) + \left(x_1^2 - x_2^2 - 1\right)$$

で与えられる。この関数は 4 個の大域的最適解を持つ。

$$\boldsymbol{x}_1 = (-\sqrt{1.5}, -\sqrt{0.5}) \quad \boldsymbol{x}_2 = (-\sqrt{1.5}, \sqrt{0.5})$$
$$\boldsymbol{x}_3 = (\sqrt{1.5}, \sqrt{0.5}) \quad \boldsymbol{x}_4 = (\sqrt{1.5}, -\sqrt{0.5})$$

参考文献

[1] A. V. Levy and A. Montalvo : "The Tunneling Algorithm for the Global Minimization of Functions," *SIAM J. Sci. Stat. Comput.*, Vol. 6, No. 1, pp. 15-29, May 1990.

[2] 三島 秀治, 佐藤 泰司, 内堀 晃彦, 田中 幹也: "適応的極配置による非線形連続系問題の解法", 電気学会電子・情報・システム部門大会講演論文集, No. B-5-3, pp. 463-464, 平成 10 年 9 月.

[3] Mou-Yan Zou and Xi Zou : "Global Optimization: An Auxiliary Cost Function Approach ," *IEEE Trans. Sys., Man and Cybern., Part A System and Humans* Vol. 30, No. 3, pp. 347-354, May 2000.