アルゴリズム　17－9
(1990. 9.28)

ランダム探索法と局所最適化法を用いた大域的最適化問題の解法について

金光 秀雄

北海道教育大学 函館分校 総合科学教室

あらまし　非線形多変数関数の最適化問題において、その局所最適解を求めるための効果的な最適化手法は数多く提案されている。しかしながら、目的関数が多峰性の場合にその大域的最適解を求めるような大域的最適化問題に対する効果的な手法は現在のところないようである。本報告では、このような大域的最適化問題を解くための手法として、ランダム探索法と局所最適化法を組み合わせた手法を紹介する。また、本手法の中で特徴的な「局所最適解と同一の峰上にある候補点を除去するための手法」について詳しく述べ、その収束性について考察する。６変数までの問題に対して本手法を適用した結果から、本手法が大域的最適化問題に対して有効であることを確かめた。

On a Method for Global Optimization Problems by Using a Random Search Method and a Local Optimization Method.

Hideo KANEMITSU

Integrated Arts and Sciences Laboratory, Hakodate College,
Hokkaido University of Education
Hakodate 040, Japan

Abstract　We propose a method for finding the optimum of global optimization problems so that the objective function has nonlinear multi-modal peaks on rectangular domains with several variables. This method uses a local optimization method. In addition, a random sampling method which picks up initial candidacy of points and a method which removes the candidacy points on the identical modal peak with the current optimum, are used for giving the effective initial point to the local optimization method. Convergence properties of this method are discussed. In numerical examples of functions with up to six variables, it is shown that this method works effectively.

1．はじめに

非線形多変数関数の最適化手法について、現在までさまざまな手法が提案されてきた。とくに、関数が単峰性の場合には、共役方向法、共役勾配法、準ニュートン法などの効果的な手法[6]が数多く提案されている。しかし、関数が多峰性で複数の局所最適解を有する場合になると、これらの手法は一つの局所最適解への収束を保証するだけで、大域的最適解への収束については初期点に依存するという大きな問題点を抱えている。

このような多峰性の関数に対して、その大域的最適解を求めるための手法は大きく三つに分類することができる。第一番目として、定義された領域上にランダムに発生させた点を利用する手法[2]などがある。しかし、これらの手法では、大域的最適解を高い精度で求めることが困難である。第二番目として、局所最適解を通過するようなパスを定め、そのパスを系統的にたどっていく手法[1][3][11]がある。しかし、これらの手法では、全域的な探索をおこなっていないため、定義したパスが全ての局所最適点を通過するという保証はない。第三番目の手法には、領域の分割を繰り返して解を含む可能性のある領域を抽出する方法[4][7][8]がある。しかし、これらの方法では、変数の数が増えるとアルゴリズムが複雑になり、計算量も急激に増大してしまうという問題が発生する。また、指数関数によるrerative peaking と多重積分を組み合わせた手法[10]は、第一番目と第三番目の手法を組み合わせた手法と考えられ、この手法が変数の多い場合にも適用でき、比較的高い精度で大域的最適解を求められる手法として知られている。いっぽう上述の一～三番目の問題点を解決するために、以下のような手順で複数の局所最適解を系統的に求める手法[5]がある。

①.全域的な探索をするランダム探索法を適用して、初期候補点を選択する。
②.候補点の中で最大の点を初期点として、局所最適化法を適用し高い精度で局所最適解を求める。
③.②で求めた局所最適解と同一の峰の上にある候補点を除去するための手法を適用して、新たなる候補点の減少を図り、再び②に戻る。

本報告では、初めに上述の手法を紹介する。つぎに本手法で特徴的な③の手法について詳しく述べ、本手法の収束性について考察する。最後にいくつかの数値例に対して本手法を適用した結果から、この手法が変数の多い大域的最適化問題にも高い精度で大域的最適解を求められることを示す。

2．準　備

2．1　問　題

本報告で扱う大域的最適化問題を定式化すると次のようになる。

$$\text{最大化} \qquad f(\mathbf{x}), \quad \mathbf{x}\in D^n\subset R^n; \quad f:D^n\rightarrow R \qquad (2.1)$$

$$\text{ただし} \qquad D^n=[L_1,U_1]\times[L_2,U_2]\times\cdots\times[L_n,U_n]$$

$L_i:x_i$の下限，$U_i:x_i$の上限

ここで、$f(\mathbf{x})$は目的関数と呼ばれ、この関数は一般に非線形で多峰性である。また、問題によっては目的関数：$f(\mathbf{x})$を最小化する場合も考えられるが、もとの目的関数に負号をつけたものを新たな目的関数として考えることにより、最大化問題に帰着できる。D^nはｎ次元空間における矩形領域をあらわしており、以後この領域を探索領域と呼び、

$$f(\mathbf{x}^{**})\leq f(\mathbf{x}), \quad \mathbf{x}\in D^n \qquad (2.2)$$

を満たす解$\mathbf{x}^{**}\in D^n$と$f(\mathbf{x}^{**})$をそれぞれ問題（２．１）の大域的最適解、大域的最適値という。本報告では最大化問題を扱うので、以後、簡単のため両者をそれぞれ最大点、最大値と呼ぶ。また、ある$\mathbf{x}^{*}\in D^n$に対して、半径$\delta>0$のｎ次元開球：$B_n(\mathbf{x}^{*};\delta)$が存在して、

$$f(\mathbf{x}^{*})\leq f(\mathbf{x}), \quad \mathbf{x}\in D^n\cap B_n(\mathbf{x}^{*};\delta) \qquad (2.3)$$

を満たす解 $\vec{x}^*$ と $f(\vec{x}^*)$ をそれぞれ問題（２．１）の局所最適解、局所最適値という。本報告では最大化問題を扱うので、以後、簡単のため両者をそれぞれ極大点、極大値と呼ぶ。したがって、条件（２．２）で定義された最大点は、条件（２．３）で定義された極大点の中に含まれる。

２．２　記　法

(1)　集合Ｓのｊ番目の要素をS_jで表す。すなわち集合Ｓが$S=\{p_1, p_2, \cdots, p_j, \cdots, p_m\}$のとき、$S_j=p_j$となる。

(2)　アルゴリズムをつぎのように表す。

（出力変数１，出力変数２，・・・，出力変数ｑ）:= アルゴリズム名（入力変数１，入力変数２，・・・，入力変数ｐ）；{* 説　明 *}

この意味は、入力変数１，入力変数２，・・・，入力変数ｐを与えてアルゴリズム名で識別されるアルゴリズムを実行すると、その結果として出力変数１，出力変数２，・・・，出力変数ｑが得られるという意味である。また、{* 説明 *} は入力変数等の説明である。特に、出力変数が１つの場合は（）を省いて表記する。

３．アルゴリズム

３．１　概　略

本手法の概略を以下に示す。

①.ランダム探索法を適用して、関数の大きい順にいくつかの初期候補点を選び出す。

②.候補点がなくなるまで③～④を繰り返す。

③.候補点の中で最大の点を初期点として、局所最適化法を適用して極大点と極大値を求め、両者を、それぞれ極大点集合と極大値集合の要素として加える。

④.②で求めた局所最適解と同一の峰上にある候補点を除去するための手法を適用して、次の繰り返しで用いる候補点の減少を図る。

以上の手順により、複数の極大点と極大値を系統的に求めることができ、この中で最大の極大値を与える極大点が、本手法を適用して求めた大域的最適化問題の解(最大点)となる。

３．２　本手法のアルゴリズム

本節では、探索領域D^n上で定義された関数に対して、その極大点集合：Ｘｅとそれぞれの点に対応する極大値集合：Ｆｅおよび極大点の個数：ｎｅを求めるアルゴリズム：Ｍｇについて述べる。

アルゴリズムＭｇ：（Ｆｅ，Ｘｅ，ｎｅ）:= ＭＧ（ｎｓ，$nc^{(1)}$，D^n，ｅｐｓ）
；{* ｎｓ:サンプル点の数，$nc^{(1)}$:初期候補点の数，D^n:探索領域，ｅｐｓ:同一解であるかどうかを判定するしきい値 *}

ｓｔｅｐ１：　繰り返し回数：ｋ、極大点集合：$Xe^{(1)}$、極大値集合：$Fe^{(1)}$、これらの添字集合：$I^{(k)}$
[初期設定]　および極大点の個数：$ne^{(1)}$を以下のように初期設定する。

（Ｓ３．２．１）　ｋ:=１；　$Xe^{(1)}:=\phi$；　$Fe^{(1)}:=\phi$；　$Ie^{(1)}:=\phi$；　$ne^{(1)}:=0$；

ｓｔｅｐ２：　一様乱数により、ｎｓ個のサンプル点：$x_i \in D^n$ (i=1, 2, ・・・, ｎｓ)を生成し、それぞれの点での関数値：$f(x_i^{(0)})$（以後$f_i^{(0)}$と記述する）を求めるアルゴリズム：Ｒｓを適用して、点の集合：$X^{(0)}$と、それぞれの点での関数値の集合：$F^{(0)}$およびこれらの添字集合：$I^{(0)}$を求める。
[ﾗﾝﾀﾞﾑ探索]

（Ｓ３．２．２）　（$X^{(0)}$，$F^{(0)}$，$I^{(0)}$）:= Ｒｓ（D^n，ｎｓ）
ただし、$I^{(0)}:=\{1, 2, \cdots, ns\}$

step3： [初期候補点の設定]　$F^{(0)}$から関数値の大きい順にnc 個取り出し、x_i, f_iを関数値の大きい順に並びかえるソーティングアルゴリズム:Scを適用して、候補点の集合:$X^{(1)}$と、それぞれの点での関数値の集合:$F^{(1)}$およびこれらの添字集合:$I^{(1)}$を求める。

(S3.2.3)　$(F^{(1)}, X^{(1)}, I^{(1)}) := Sc(F^{(0)}, X^{(0)}, I^{(0)}, nc^{(1)})$

ただし、$I^{(1)} := \{1, 2, \cdots, nc^{(1)}\}$ および、

任意の$i<j \in I^{(1)}$ に対して、$f_i^{(1)} \geqq f_j^{(1)}$

step4： [局所最適化]　step3で求めた$X^{(k)}$, $F^{(k)}$の中で最も大きな関数値:f_m を与える点:x_mを求め、この点およびこの点での関数値を$X^{(k)}$, $F^{(k)}$から除く。次に、この点を初期点として局所最適解を求めるアルゴリズム:MLを適用して、極大点:$\dot{x}$ およびその点での関数値:$\dot{f}$を求める。

(S3.2.4)　$m := \{i \mid \max(f_i), i \in I^{(k)}\}$;

(S3.2.5)　$X^{(k)} := X^{(k)} - \{x_m\}$;　$F^{(k)} := F^{(k)} - \{f_m\}$;

(S3.2.6)　$(\dot{f}, \dot{x}) := ML(f_m, x_m, eps)$;

step5： [収束判定]　step4で求めた$\dot{x}$ が$Xe^{(k)}$の極大点と同一であるかどうかを判定するアルゴリズム:Ecを適用して、新たな極大点集合:$Xe^{(k+1)}$、極大値集合:$F^{(k+1)}$および、極大値の個数:$ne^{(k+1)}$を求める。また、このとき候補点の集合:$X^{(k)}$が空ならば、本アルゴリズムを終了する。さらに、step4で求めた極大点:$\dot{x}$ が極大点集合:$Xe^{(k)}$ の中の極大点と同一である時には、現在の候補点の集合、この点での関数値の集合、添字の集合および候補点の個数を、それぞれ次の繰り返しの集合とし、繰り返し回数を1つ増やしてstep4に戻る。

(S3.2.7)　$(Fe^{(k+1)}, Xe^{(k+1)}, Ie^{(k+1)}, ne^{(k+1)}, same\_sol) := Ec(Fe^{(k)}, Xe^{(k)}, Ie^{(k)}, ne^{(k)}, \dot{x}, eps)$;

{* same_sol=TRUE: 同一極大点有り、same_sol=FALSE: 同一極大点無し *}

(S3.2.8)　if $X^{(k)} = \Phi$ then end of MG;

(S3.2.9)　if same_sol = TRUE then

(S3.2.10)　$F^{(k+1)} := F^{(k)}$;　$X^{(k+1)} := X^{(k)}$;　$I^{(K+1)} := I^{(k)}$;　$nc^{(K+1)} := nc^{(k)}$;

(S3.2.11)　k := k + 1 ;　goto step4;

step6： [候補点除去]　現在の候補点集合:$X^{(k)}$の各点の中で、step4で求めた極大点:$\dot{x}$と同一の峰上にある点を除去するアルゴリズム:Rimを適用して、新たな候補点集合:$X^{(k+1)}$、関数値の集合:$F^{(k+1)}$、添字集合:$I^{(k+1)}$および候補点の個数:$nc^{(k+1)}$ を求め、繰り返し回数を1つ増やしてstep4に戻る。

(S3.2.12)　$(F^{(k+1)}, X^{(k+1)}, I^{(k+1)}, nc^{(k+1)}) := Rim(F^{(k)}, X^{(k)}, I^{(k)}, nc^{(k)}, \dot{f}, \dot{x})$;

(S3.2.13)　k := k + 1 ;　goto step4

4. 極大点と同一の峰上にある候補点の除去

3章で示した、全体のアルゴリズムのstep6において、候補点を除去するアルゴリズムがある。このアルゴリズムは、現在までに得られている極大点に再び収束する可能性のある点を除去し、次の繰り返しで実行する局所最適化法に対して効果的な初期点を与えるという重要な役割を果たしており、本手法の全体の性能に大きく関わる部分である。すなわち、もしこのアルゴリズムを用いない場合には、ある極大点と同一の峰上にある候補点全てに対して、局所最適化法を適用することになり、これらの点はすべてが同じ極大点に収束してしまうため、はなはだ効率の悪い手法となっ

てしまう。

ここでは、まず候補点を除去するアルゴリズムを示す。つぎに、その中で特に重要である候補点が極大点と同一の峰上にあるかどうかを判定するアルゴリズムについて詳しく述べる。

4．1　候補点の除去アルゴリズム

本節では、次の繰り返しで用いる候補点集合：$X^{(k+1)}$の個数：$nc^{(k+1)}$の減少を図るために、局所最適化手法で求めた局所最適解：$\dot{x}$と同一の峰上にある候補点：$X_j^{(k)}$ を除去するためのアルゴリズム：Rim について述べる。

アルゴリズムRim：$(F^{(k+1)}, X^{(k+1)}, I^{(k+1)}, nc^{(k+1)}) := Rim(F^{(k)}, X^{(k)}, I^{(k)}, nc^{(k)}, \dot{f}, \dot{x})$；
{＊ $F^{(k)}$：候補点での関数値集合、$X^{(k)}$：候補点集合、$I^{(k)}$：添字集合、$nc^{(k)}$：候補点の数、$\dot{f}$：極大値、$\dot{x}$：極大点　k：繰り返し回数　＊}

step1：[初期設定]　つぎの繰り返しで用いる各集合および候補点の個数を、それぞれ空集合と0に初期設定する。

(S4.1.1)　$F^{(k+1)} := \phi$；　$X^{(k+1)} := \phi$；　$I^{(k+1)} := \phi$；　$nc^{(k+1)} := 0$；

step2：[候補点の除去]　jを1から現在の候補点の個数$nc^{(k)}$まで変えながら、現在の候補点：$X_j^{(k)}$、極大点：$\dot{x}$およびそれぞれの点での関数値を与えて、極大点と同一の峰上にあるかを判定するアルゴリズム：Jimを適用し、同一の峰上にない候補点：$X_j^{(k)}$だけを次の繰り返しの候補点集合：$X^{(k+1)}$に加える。

(S4.1.2)　for　j:=1　to　$nc^{(k)}$　do

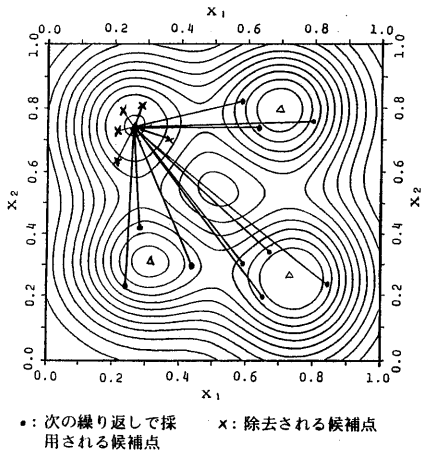
(S4.1.3)　idntfy_modal := Jim($F_j^{(k)}$, $X_j^{(k)}$, $\dot{f}$, $\dot{x}$)；
{＊ idntfy_modal＝TRUE：極大点と候補点が同一の峰上にある
idntfy_modal＝FALSE：極大点と候補点が同一の峰上にない　＊}

(S4.1.4)　if　idntfy_modal＝FALSE　then

(S4.1.5)　$nc^{(k+1)} := nc^{(k+1)} + 1$；　$I^{(k+1)} := I^{(k+1)} + \{nc^{(k+1)}\}$；

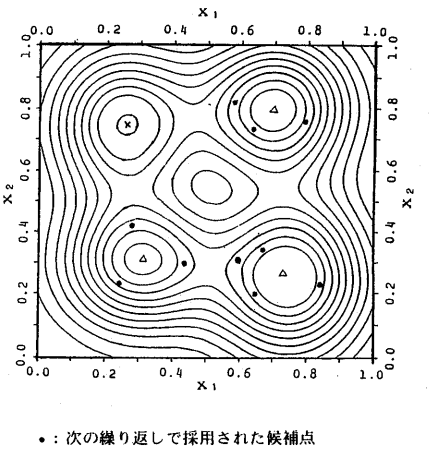
(S4.1.6)　$X^{(k+1)} := X^{(k+1)} + \{X_j^{(k)}\}$；　$F^{(k+1)} := F^{(k+1)} + \{F_j^{(k)}\}$；

このアルゴリズムでは、(S4.1.1)で、次の繰り返しで用いる各集合を空集合に初期設定してし、現在の各集合の要素の中で、極大点と候補点が同一の峰上にないときだけ、(S4.1.5)～(S4.1.6)で、その要素を次の繰り返しで用いる各集合に加えている。そのため、極大点と候補点が同一の峰上に場合の個数分だけ、現在の候補点が除去されることになる（図4.1、図4.2）。



●：次の繰り返しで採用される候補点　×：除去される候補点

図4．1　候補点除去アルゴリズムを適用している状態



●：次の繰り返しで採用された候補点

図4．2　候補点除去アルゴリズムを適用後の状態

４．２　候補点が極大点と同一の峰上にあるかを判定するアルゴリズム

本節では、候補点:$X^{(k)}$と極大点:$\dot{x}$ が同一の峰上にあるかどうかを判定し、同一の峰上にある場合には idntfy_modal:=TRUE に、同一の峰上にない場合には idntfy_modal:=FALSE に設定するアルゴリズム:Ｊim （以後、同一峰判定アルゴリズムと呼ぶ）について述べる。

アルゴリズムＪim:　idntfy_modal := Ｊim（$F_j^{(k)}$, $X_j^{(k)}$, f, $\dot{x}$）;

{* $F_j^{(k)}$:ｊ番目の候補点での関数値、$X_j^{(k)}$:ｊ番目の候補点、f:極大値、$\dot{x}$:局所最適解 *}

ｓｔｅｐ１:　同一峰判定フラグ:idntfy_modal:=TRUE に、直線最小化する方向:d_jを候補点[初期設定]　:$X^{(k)}$と極小点$\dot{x}$の二点間を結ぶ方向に初期設定する。

(Ｓ４．２．１)　idntfy_modal := TRUE　;　$d_j := X_j^{(k)} - \dot{x}$　;

ｓｔｅｐ２:　極大点:$\dot{x}$から方向:d_jに以下のような大域的な最小化をおこない、

$$\phi(\alpha) = f(\dot{x} + \alpha d_j)$$

としたとき、$f\dot{m}c = \min(\phi(\alpha), \alpha \in [0,1])$　（４．１）

この手続きの一回の繰り返し毎に評価した関数値:fmc、およびこの手続きが終了したかを示すフラグ:end を出力するアルゴリズム:Ｌmlit をidntfy_modal＝TRUEかつ end＝FALSEの間繰り返して適用する。この間、もし、ここで得られた関数値:fmc が候補点での関数値 $F_j^{(k)}$より小さければ、idntfy_modal:=FALSE とする。

(Ｓ４．２．２)　**ｒｅｐｅａｔ**

(Ｓ４．２．３)　(fmc, end) := Ｌmlit（f, $\dot{x}$, $F_j^{(k)}$, $X_j^{(k)}$, d_j）;

{* end＝TRUE/FALSE: 直線最小化の手続きが 終了した/していない *}

(Ｓ４．２．４)　**ｉｆ**　fmc < $F_j^{(k)}$　**ｔｈｅｎ**

(Ｓ４．２．５)　idntfy_modal := FALSE;

(Ｓ４．２．６)　**ｕｎｔｉｌ**　(idntfy_modal＝TRUE かつ end＝FALSE) ;

本手法の意味について、考察する。つぎのようなレベル集合、

$$L(fmc) = \{x \mid f(x) \geqq fmc\}$$　（４．２）

を考えたとき、$fmc \geqq F_j^{(k)}$が成立すれば、そのレベル集合:L(fmc) は、$\dot{x}$と同一の峰上にあるという可能性が高い。したがって、もし次の繰り返しでこの候補点を初期点として局所最適化手法を適用しても、求められた極大点は現在の極大点:$\dot{x}$に収束してしまう可能性が高いので、このような候補点:$X_j^{(k)}$を除去するのが妥当である。一方、$fmc < F_j^{(k)}$が成立しているときは、$\dot{x}$と異なる峰上 にある可能性が高いため、このような候補点:$X_j^{(k)}$は残しておくべきである（図４．３）。

以上の手続きにより、同一の峰上にある候補点の除去アルゴリズムは、極大点と候補点との間での一方向大域的最小化アルゴリズムで構成できることが示された。しかしながら、このような正確

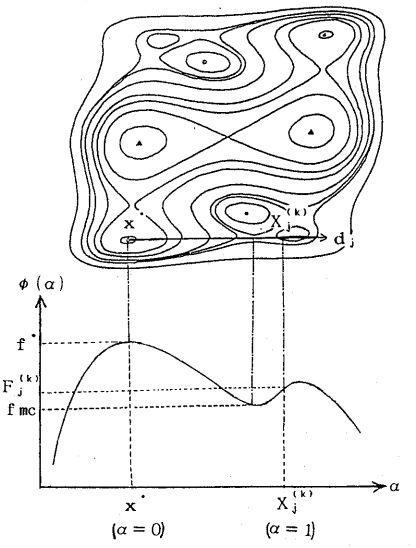


図４．３　同一峰判定アルゴリズム

な一方向大域的最小化アルゴリズムは、一般に無限の手続きを要するので現実的ではない。そこで、多少確実性に欠けるが、実用的なアルゴリズム:Jima を、同一峰上判定アルゴリズムに含めたかたちで示していく。

アルゴリズムJima:　idntfy_modal := Jima ($F_j^{(k)}$, $X_j^{(k)}$, f, $\dot{x}$) ;

{* $F_j^{(k)}$:j番目の候補点での関数値、$X_j^{(k)}$:j番目の候補点、f:極大値、$\dot{x}$:局所最適解 *}

step1:　最小化方向:d_jを候補点$X^{(k)}$と極小点:$\dot{x}$の二点間を結ぶ方向に初期設定する。

(S4.2.7)　$d_j := X_j^{(k)} - \dot{x}$;

step2:　極大点:$\dot{x}$から方向:d_jに以下のような近似的な最小化をおこない、同一峰判定フラグを設定する。

$\phi(\alpha) = f(\dot{x} + \alpha d_j)$とおくと、極大値:$f^*$は$\phi(0)$、候補点での関数値:$F_j^{(k)}$は$\phi(1)$となる。また二点の中間の点での関数値:$\phi(0.5)$を計算する。ここで、$\phi'(0) = 0$ に注意すると、この3点で、3次の関数近似ができ、このときの最小点の予測値α^*は、

$$\alpha^* = (8P-1)/3(4P-1), \quad \text{ただし} P = (\phi(0.5) - \phi(0))/(\phi(0.5) - \phi(0)) \qquad (4.3)$$

となる。よって、以下のようなステップを構成できる。

(S4.2.8)
```
if  0 < α* < 1 かつ φ(α*) < φ(1)
    then  idntfy_modal := FALSE;
    else  idntfy_modal := TRUE;
```

5. 収束性

[定理5.1]　アルゴリズムMg は有限回の繰り返し回数:k ($\leq nc^{(1)}$) で必ず停止する。

(証明)　このアルゴリズムにおいて、(S3.2.5)は、各繰り返し毎に必ず実行される。従って、各繰り返し毎に $X^{(k+1)} \subset X^{(k)}$となり、$X^{(1)}$の要素の個数は$nc^{(1)}$であるから、$k \leq nc^{(1)}$で$X^{(1)} = \phi$となる。よって、停止条件(S3.2.8)より、アルゴリズムMg は有限回の繰り返し回数:k ($\leq nc^{(1)}$) で必ず停止する。　(証明終)

[定理5.2]　本手法において、次の条件。

(1) 極大点:$\dot{x}$に対して同一峰判定アルゴリズム:Jimを適用して idntfy_modal=TRUE となった点を初期点として、局所最適化アルゴリズム:MLを適用すると、常に現在求められている極大点:$\dot{x}$に収束する。

(2) あるj$\in I^{(1)}$が存在して、この候補点:$X_j^{(1)}$を初期点として、局所最適化アルゴリズム:MLを適用すると、大域的最適解:$\dot{\dot{x}}$に収束する。

を満足すれば、本手法により大域的最適解:$\dot{\dot{x}} \in Xe$ を求めることができる。

(証明)　(1),(2)の条件のもとで、本手法を適用して求めたあらゆる極大点が大域的最適解でないと仮定して、矛盾を導く。この仮定と条件(2)および(S3.2.12)より、ある繰り返しk=k1において、$X_{j1}^{(K1)} = X_j^{(1)}$, (j$\in I^{(1)}$) 、$X_{j1}^{(K1)} \in X^{(K1)}$、$X_{j1}^{(K1)} \notin X^{(K1+1)}$なる候補点:$X_{j1}^{(K1)}$が存在し、この点を初期点として、局所最適化アルゴリズム:MLを適用すると大域的最適解:$\dot{\dot{x}}$に収束させることができる。またこのような点:$X_{j1}^{(K1)}$では(S4.1.4)～(S4.1.6)より、idntfy_moda

1＝TRUEである。したがって、条件(1)より、この点は現在求められている極大値:$\dot{x}$に収束する。すなわち、現在求められている極大値:$\dot{x}$は大域的最適解:$\ddot{x}$であることになり、最初の仮定に反し、矛盾する。 (証明終)

6．数値例

(1) 関数（２変数、６極大点）:Six Hump Camel Back Function[1]

$$f(x) = -4x_1^2 + 2.1x_1^4 \frac{1}{3} x_1^6 - x_1x_2 + 4x_2^2 - 4x_2^4$$

探索領域　$-2.0 \leqq x_1 \leqq 2.5$,　　$-1.0 \leqq x_2 \leqq 1.5$

最大点 : $\ddot{x} = (-0.0898, 0.7126)$, $\ddot{x} = (0.0898, -0.7126)$,　最大値 : $f(\ddot{x}) = 1.0316$

結果

番号 *はglobal	極大点		極大値
	x_1	x_2	$f(\dot{x})$
1*	**0.089842**	**-0.712656**	**1.031628**
2*	**-0.089842**	**0.712657**	**1.031628**
3	-1.703607	0.796084	0.2154675
4	1.703600	-0.796088	-0.2154675

項目名	関数評価回数	計算時間
全体	839回	0.097秒
ランダム探索(初期候補点:100)	300回(36%)	0.044秒(45%)
局所最適化(6手続き)	250回(30%)	0.032秒(33%)
候補点除去(4手続き)	289回(34%)	0.006秒(6%)
その他		0.015秒(15%)

(2) 関数（５変数、８極大点）[10]

$f_1(x_1) = x_1(x_1+13)(x_1-15) \times 0.01$; $f_2(x_2) = (x_2+15)(x_2+1)(x_2-8) \times 0.01$

$f_3(x_3) = (x_3+9)(x_3-2)(x_3-9) \times 0.01$; $f_4(x_4) = (x_4+11)(x_4+5)(x_4-9) \times 0.01$

$f_5(x_5) = (x_5+9)(x_5-9)(x_5-10) \times 0.01$

$f(x) = f_1(x_1) \times f_2(x_2) \times f_3(x_3) \times f_4(x_4) \times f_5(x_5)$

探索領域　$-10 \leqq x_1, x_2, x_3, x_4, x_5, x_6 \leqq 10$

最大点 : $\ddot{x} = (8.7564, -9.35826, -4.5720, 3.5921, -2.8400)$,　最大値 : $f(\ddot{x}) = 24416.03$

結果

番号 *はglobal	極大点					極大値
	x_1	x_2	x_3	x_4	x_5	$f(\dot{x})$
1*	**8.756444**	**-9.358201**	**-4.572125**	**3.592080**	**-2.839987**	**24416.03**
2	8.756445	10.00000	-4.572085	3.592131	-2.840091	16405.84
3	-7.423104	4.024964	-4.572073	3.592144	-2.840089	8846.346
4	-7.423110	-9.358304	5.905385	3.592112	-2.840104	8690.922
5	-7.422360	-9.357983	-4.572077	10.00000	-2.840176	8852.502
6	8.756735	-9.358207	10.00000	3.592125	-2.840058	9396.575
7	8.757124	4.024949	5.905412	3.592127	-2.840110	5170.143
8	-7.423303	-9.357942	-4.572360	-8.258829	-2.840319	4332.759
9	8.757191	4.024466	-10.00000	3.592080	-2.840127	6543.704

10	-7.434753	-9.357213	-10.00000	3.590143	-2.838137	10999.81
11	8.756449	4.024952	-4.571968	10.00000	-2.840119	5266.262
12	-7.423318	10.00000	5.905544	3.592436	-2.839498	5839.738
13	8.756464	-9.358272	5.905409	10.00000	-2.840081	5173.741
14	8.757682	-9.357650	5.905410	3.578329	-10.00000	4518.806
15	8.756437	-9.358308	5.905422	3.592106	-2.840085	2532.229
16	-7.422915	4.024938,	10.00000	3.592106	-2.840085	3404.540
17	-7.423107	-9.357716	-4.572132	-3.592820	-2.840053	7731.919
18	8.756713	4.024959	-4.572087	-8.258805	-2.840053	2577.514
19	8.756489	-9.358220	-10.00000	-8.258797	-2.840080	3204.971
20	-7.425801	10.00000	-4.572107	-8.258796	-2.839859	2911.333

項　目　名	関数評価回数	計 算 時 間
全　体	9351回	3.360秒
ランダム探索(初期候補点:480)	3400回(36%)	2.625秒(78%)
局所最適化(23手続き)	2714回(29%)	0.338秒(10%)
候補点除去(20手続き)	3237回(35%)	0.073秒(2%)
その他		0.324秒(10%)

(3) 関数 (6変数、8極大点) [9]

$y1=-5\{x_1^2+(x_2^2-1)\}$; $y2=-1\{(x_1^2-1)+x_2^2\}$; $y3=-5\{x_3^2+(x_4^2-1)\}$

$y4=-1\{(x_3^2-1)+x_4^2\}$; $y5=-5\{x_5^2+(x_6^2-1)\}$; $y6=-1\{(x_5^2-1)+x_6^2\}$

$f_1=e^{y1}+e^{y2}$; $f_2=e^{y3}+e^{y4}$; $f_3=e^{y5}+e^{y6}$; $f(x)=f_1\times f_2\times f_3$

探索領域 $-0.5\leqq x_1, x_2, x_3, x_4, x_5, x_6\leqq 1.5$

最大点 : $\vec{x}^{**}=(0.0297, 0.9702, 0.0297, 0.9702, 0.0297, 0.9702)$; 最大値 : $f(\vec{x}^{**})=1.49466$

結果

番号 *はglobal	極大点						極大値
	x_1	x_2	x_3	x_4	x_5	x_6	$f(x^*)$
1*	**0.029795**	**0.970203**	**0.029795**	**0.970205**	**0.029796**	**0.970205**	**1.494668**
2	0.029795	0.970201	0.029798	0.970206	0.999782	0.000231	1.307323
3	0.999773	0.000228	0.999772	0.000228	0.029795	0.970205	1.143460
4	0.999751	0.000239	0.029798	0.970205	0.999774	0.000229	1.143460
5	0.029794	0.970205	0.999772	0.000228	0.029795	0.970205	1.307323
6	0.029858	0.970200	0.999772	0.000234	0.999772	0.000257	1.143460
7	0.999772	0.000228	0.999772	0.000230	0.999772	0.000228	1.000137
8	0.999772	0.000228	0.029795	0.970204	0.000298	0.970205	1.307323

項　目　名	関数評価回数	計 算 時 間
全　体	10999回	7.265秒
ランダム探索(初期候補点:500)	8300回(75%)	6.985秒(96.1%)
局所最適化(8手続き)	715回(7%)	0.101秒(1.4%)
候補点除去(8手続き)	1984回(18%)	0.065秒(0.9%)
その他		0.115秒(1.6%)

(1) の数値例では、6個の極大点のうち4個の極大点が求められているが、残り2個の極大点はその極大値($f^*=-2.1042$)が小さすぎ、初期候補点に採用されなかったためである。(2)の数値例では、文献[2]の方法によると、21000回の関数評価回数で最大点を1個求めているが、本手法では、9351回の関数評価回数で20個の極大点(1個:最大点)を非常に高い精度で求めることができた。(3)の数値例では、6変数の問題であるが、全ての極大点が非常に高い精度で求めることができた。尚、これらの計算には、北海道大学・大型計算機センターの HITAC M-682H を利用した。

7. おわりに

本報告で得られた結果を要約する。

(1) ランダム探索法と局所最適化法の組み合わせたアルゴリズムを示した。

(2) 本手法の中で特徴的な極大値と同一の峰上にある候補点を除去するためのアルゴリズムを示し、これが最終的に一方向大域的最小化アルゴリズムを用いて構成できることを示した。

(3) 本手法の収束性について考察し、この手法が初期候補点の数を上限とする繰り返し回数で停止すること示すとともに、本手法が大域的最適解に収束する条件を示した。

(4) いくつかの数値例に対して本手法を適用した結果から、この手法が変数の多い大域的最適化問題に対しても、高い精度で大域的最適解を求められることを示した。

今後の課題としては、

(1) 効果的な初期候補点の数を与えるための指針について検討する。

(2) 本手法に必要な計算量の上限と下限について検討する。

(3) ランダム探索法の計算の効率化を図る。

が考えられる。

謝辞 本研究を進めるにあたり、日頃よりご指導していただいている北海道大学 新保勝教授、宮腰政明助教授、ならびに大変有益なご助言を頂いた北海道大学 伊達惇教授に深く感謝致します。

参考文献

[1] F.H.Branin Jr.: Widely Convergent Methods for Finding Multiple Solutions of Simultaneous Nonlinear Equations. IBM J.Res.Develop., Sept.(1972), pp.504-522.

[2] S.H.Brooks: A Discussion of Random Methods for Seeking Maxima. Operations Research, Vol.6, No.2(1958), pp.244-251.

[3] 榎本, 片山, 川本: リンク手法による多変数関数の極値探索. 情報処理, Vol.17, No.9(1976), pp.835-842.

[4] 藤井, 市田, 清野: 区間演算を利用した多変数関数の最大値探索法. 情報処理, Vol.18,No.11(1977), pp.1095-1101.

[5] 金光: 非線形多変数関数の最適化アルゴリズムに関する研究. 北海道大学大学院工学研究科情報工学専攻修士論文, 1982.

[6] 今野, 山下: 非線形計画法. 日科技連(1978).

[7] 水野: 分枝限定法を用いた方程式の解法. 日本オペレーションズ・リサーチ学会論文誌, Vol.30, NO.30(1987), pp.41-58.

[8] 正道寺: 多変数多峰性関数に関する最適値探索法. 日本オペレーションズ・リサーチ学会論文誌, Vol.20, No.4(1977), pp.311-320.

[9] 津田: モンテカルロ法とシミュレーション(改訂版). 培風館(1977).

[10]津田, 佐藤: 多峰性多変数関数の最大・最小. 情報処理, Vol.16, No.1(1975), pp.2-6.

[11] H.Yamashita: A Differential Equation Approach to Nonlinear Programming.Mathematical Programming 18(1980), pp.155-168.