[計測自動制御学会論文集]
Vol. 29, No. 11, 1342/1351 (1993)

徐冷型ランダム・トンネリング・アルゴリズムによる大域最適化†

島　　孝　司*

Global Optimization by Annealing Type of Random Tunneling Algorithm

Takashi SHIMA*

We will propose new global optimization procedure which utilizes the concept of tunneling algorithm in this paper. There exist two phases in the tunneling algorithm for global minimization problems, 1) minimization phase and 2) tunneling phase. In the minimization phase, the local minimum is searched. In the tunneling phase, the point in the lower valley is searched.

In this paper, we will propose an annealing type of random tunneling search in the tunneling phase. First, we set the temperature at some degree and generate the increment (δx) from the local minimum (x^*), according to the Cauchy distribution. A new point $x(=x^*+\delta x)$ is evaluated whether it is in the lower valley or not. If the point in the lower valley is not found after the predetermined number of trials, the temperature is set a little lower according to its cooling schedule and we repeat the same procedure again.

The wide range of search becomes possible owing to the property of Cauchy distribution and temperature cooling. We also abopt multistart scheme in order not to miss the global minimum.

Several test problems are solved. The proposed method hardly misses the global minimum and is very promising for future application research.

Key Words: global optimization, random tunneling algorithm, Cauchy distribution, multistart scheme, annealing

1. はじめに

近年，シミュレーテッド・アニーリング，遺伝アルゴリズムなどの新しい最適化のアルゴリズムが提案され，大域最適化の研究が非常に活発に行われている．本稿では，トンネリングアルゴリズムの概念を応用した大域最適化手法を提案する

トンネリングアルゴリズムの概念は非常に簡単なものである[1),2),5),8)]．たとえば **Fig. 1** に示すような微分可能関数の最小化問題を考えよう．興味のある，対象とする範囲にランダムに初期値を発生する．もし，図の A 点が初期値になったとすると，勾配法やそのほかの既存の非線形最適化手法で，大域最適解になる G 点をみつけること

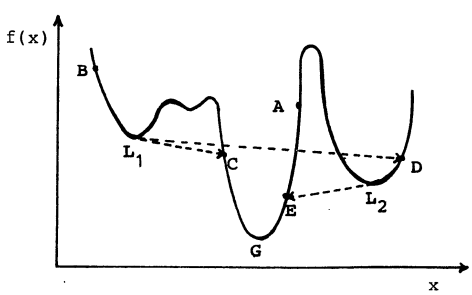


Fig. 1 Conceptual figure of tunneling algorithm

† 第18回システムシンポジウム，第16回知能システムシンポジウム，第2回ニューラルネットワークシンポジウム，合同シンポジウムで一部発表 (1992・9)

* 金沢女子短期大学　金沢市末町10
* Kanazawa Women's College, Kanazawa

(Received August 11, 1992)
(Revised March 25, 1993)

ができる．これに対して，もし図の B 点が初期値となると，上述の方法では局所最適解 L_1 点に収束してしまう．トンネリングアルゴリズムでは局所解 L_1 から脱出するために，あたかもトンネルを掘るごとく L_1 が極値をとる谷よりも深い谷，つまり L_1 のもつ目的関数値よりも改良された目的関数値をもつ点を探索する．たとえば，C 点がみつかったとする．C 点を初期値として上述の方法を再度適用することによって G 点をみつけることができる．もし，L_1 から D 点が求められたとする．D 点を初期値とすると局所解 L_2 に収束する．再度，L_2 よりもより深い谷を探索し，もし E 点がみつかれば，大域最適解に対応する G 点に収束することになる．

さて，探索途上で G 点を簡単に大域最適解というのは難しい．よって実際には G 点よりもより深い谷が存在するかどうか探索し，より深い谷が存在しなければ，G 点よりもより改良された目的関数値をもつ点が存在しないとして，G 点を大域最適解とするということである．

上からもわかるように，トンネリングアルゴリズムは，局所解探索のフェーズと，より深い谷をみつけるためのトンネリングのフェーズに分けられる．一般的には，minimization system，tunneling system[8]などと呼ばれているが，本稿では1）最小化のフェーズ，2）トンネリングのフェーズ，と呼ぶことにする．

トンネリングアルゴリズムと呼ばれるものは文献2）において提唱されている．そこでは，局所解 $\boldsymbol{x}^*$ を脱出するために，つぎのようなトンネリング関数を利用している．

$$h(\boldsymbol{x})=\frac{f(\boldsymbol{x})-f^*}{[(\boldsymbol{x}-\boldsymbol{x}^{1*})^t(\boldsymbol{x}-\boldsymbol{x}^{1*})]^{\eta_1}\cdots[(\boldsymbol{x}-\boldsymbol{x}^{m*})^t(\boldsymbol{x}-\boldsymbol{x}^{m*})]^{\eta_m}}=0 \qquad (1)$$

ここで，$f(\boldsymbol{x})$ は目的関数，$\boldsymbol{x}$ は決定変数，$\boldsymbol{x}^{i*}$，$i=1,2,\cdots,m-1$ はこれまでにみつけられた局所解，$\boldsymbol{x}^{m*}=\boldsymbol{x}^*$ は直前にみつけられた目的関数値が f^* である局所解，η_i は $\boldsymbol{x}^{i*}$ が $h(\boldsymbol{x})$ の極になるような十分大きな数である．

ほかの谷に属する点 $\boldsymbol{x}$ は（1）式の零点（根）を求めることにより得られる（よってこの場合，直前にみつかった局所解と同じ目的関数値をもつ，まだ未発見の谷の点を探索することになる．つまり，未発見の谷へ向け水平にトンネルを掘ることになる）．しかし，文献8）にもあるように，この関数の根をみつけることはかなり難しく，なによりも局所解の個数が増えるとそれだけで（1）式は非常に複雑になってしまうという欠点がある．

これを解消するために，つぎのような微分方程式を解いて，より深い谷の点 $\boldsymbol{x}$ を求める，ダイナミック・トンネリング・アルゴリズムと呼ばれる方法が文献8）で提案された．

$$\frac{dx_i}{dt}=-\frac{\dfrac{\partial f}{\partial x_i}}{[(\boldsymbol{x}-\boldsymbol{x}^*)^t(\boldsymbol{x}-\boldsymbol{x}^*)]}-\sum_{j=1}^{m}k_j f^*(g_j(\boldsymbol{x}))\frac{\partial g_j(\boldsymbol{x})}{\partial \boldsymbol{x}_i}-kf^*(\hat{f}(\boldsymbol{x}))\frac{\partial f}{\partial \boldsymbol{x}_i} \qquad (2)$$

$$(i=1,2,\cdots,n)$$

ここで，$f(\boldsymbol{x})$ は目的関数，$g_j(\boldsymbol{x})$ は制約式，

$f^*(z)=z$ if $z>0$，$f^*(z)=0$ if $z\leqq 0$，

$\hat{f}(\boldsymbol{x})=f(\boldsymbol{x})-f(\boldsymbol{x}^*)$，

t は転値を表わす．k，k_j は係数．

ダイナミック・トンネリング・アルゴリズムでは，局所解 $\boldsymbol{x}^*$ を脱出するため $\boldsymbol{x}^*\pm\boldsymbol{\varepsilon}$（$\boldsymbol{\varepsilon}$ は非常に小さな正数）を初期値としてルンゲ・クッタ法などで（2）式の微分方程式を解く．しかし，経験的には，k などの係数の調整が非常に難しく，係数の値によっては，より浅い谷の点に捕まることもあり，大域最適解を得るためには，多大な努力が必要である．

いずれにしても，これまでのトンネリングアルゴリズムは，さまざまな問題を抱えているといえよう．

本稿では，これまでのトンネリングアルゴリズムとは全く異なる，新しいアルゴリズムを提案する．大域最適化のテーマは局所解からの脱出であるが，本稿で提案する手法も，最小化のフェーズには問題はなく，主要箇所はトンネリングのフェーズである．本研究では，トンネリングのフェーズにランダム現象を利用した手法を考える．局所解からの増分を Cauchy 分布に従って発生し，温度係数の冷却スケジュールを利用して，広くかつシステマティックに決定変数の空間を探索し，局所解から脱出する徐冷型のランダムトンネリングアルゴリズムを提案する．数値例に使用した問題に対して，非常に良い結果が得られており，テスト関数の大域最適解をほぼ逃すことなくみつけることができている．以下のセクションでこの方法について紹介する．

2. 徐冷型ランダム・トンネリング・アルゴリズムについて

2.1 アルゴリズムとその概要について

徐冷型ランダム・トンネリング・アルゴリズムの詳細は後に述べることにして，まずアルゴリズムの概要を述べ，つぎにアルゴリズムを紹介する．

本稿ではつぎのような最適化問題を考えることにする．

$$\min f(\boldsymbol{x}) \quad \text{subject to } \boldsymbol{x}\in X \qquad (3)$$

ここで，$\boldsymbol{x}$ は n 次元決定変数，$f(\boldsymbol{x})$ は極値が孤立点であ

る多峰性関数で，簡単のため本稿では微分可能であるとする．X は実数空間 R^n 全体，またはその部分空間である n 次元超立方体であるが，簡単のため当分の間は R^n 全体であるとする．

第1章のセクションでトンネリングアルゴリズムを簡単に紹介したが，本研究での徐冷型ランダム・トンネリング・アルゴリズムも全く同様の構造をもっている．

まず，興味ある，対象とする領域にランダムに初期値 $\boldsymbol{x}_0$ を発生する．この初期値 $\boldsymbol{x}_0$ をもとに，勾配法など既存の非線形最適化手法で解 $\boldsymbol{x}^*$ をみつける．多くの場合，この解は局所解であるが，大域最適解である場合もある．以上が最小化のフェーズである．本手法では，適切でありさえすれば，最小化のフェーズにどのような非線形最適化手法を用いてもかまわない．後に述べる説明の都合上，本稿では，勾配法(最急降下法：ただし，簡単のためステップ幅を固定する)によって説明してゆくが，これはあくまで説明のための便宜上のことで，勾配法を使わねばならないといったことは全くない．むしろ本格的には，ほかの精度の高い方法を使用したほうが良いと思われる．つぎに，トンネリングのフェーズへ移る．

トンネリングのフェーズではみつかった局所解 $\boldsymbol{x}^*$ からの増分 $\boldsymbol{\delta x}$ を以下の式を使いランダムに発生する．

$$\delta x_i = T \tan P_i \quad (i=1, \cdots n) \tag{4}$$

後に詳細に説明するように，この式はファースト・シミュレーテッド・アニーリング法(以下 FSA)[6]に使用される式で，Cauchy 分布に従う式である．T は係数であるが，FSA に従って温度と呼ぶことにする．本法ではこの T を徐々に小さくしながら探索を行う．P_i は一様乱数を $(-\pi/2, \pi/2)$ 間の数に変換した数である．

まず，T をある値に設定する．各次元を独立とし，δx_i (すなわち P_i)をランダムに発生し，$\boldsymbol{\delta x}$ を作る．新しい点 $\boldsymbol{x}$ を

$$\boldsymbol{x} = \boldsymbol{x}^* + \boldsymbol{\delta x} \tag{5}$$

により発生する．もし

$$f(\boldsymbol{x}) < f(\boldsymbol{x}^*) \tag{6}$$

ならば，より深い谷の点がみつかったわけであるから最小化のフェーズへ帰り，みつかった点 $\boldsymbol{x}$ を初期値として最小化のフェーズを実行し，つぎの局所解を求める．もし，$\boldsymbol{x}$ が(6)式を満たさなければ，再度 $\boldsymbol{\delta x}$ を発生して新しく評価する．これをあらかじめ決められた回数(ITERS 回)だけ実行してみつからなければ，T を少し小さくして，再度同じ手順を実行する．T をあらかじめ決められたスケジュールにより順次変更し(冷却し)，同じ手順を繰り返す．スケジュールが終了した時点で，より深い谷の点がみつからなければ，$\boldsymbol{x}^*$ を大域最適解とみなす．以上が簡単な本手法の概要であり，トンネリングアルゴリズムの概念を忠実に実行していることがわかる．これをまとめると以下のようになる．

徐冷型ランダム・トンネリング・アルゴリズム

(最小化のフェーズ)

Step 1：乱数の種を設定し，初期値 $\boldsymbol{x}_0$ を興味ある，対象とする範囲に発生する．

Step 2：$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_0 - \alpha \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\boldsymbol{x}_0} \tag{7}$$

により，$\boldsymbol{x}_{\text{new}}$を発生する．

Step 3：もし $|\partial f/\partial x_i|_{\boldsymbol{x}=\boldsymbol{x}_{\text{new}}} < \delta$ (8.a)

$(i=1, 2, \cdots, n)$

あるいは

$$\|\partial f/\partial \boldsymbol{x}|_{\boldsymbol{x}=\boldsymbol{x}_{\text{new}}}\| < \omega \tag{8.b}$$

(δ, ω は非常に小さな正数)

であれば局所解に収束したとし，$\boldsymbol{x}^* = \boldsymbol{x}_{\text{new}}$ として，トンネリングフェーズ(Step 4)へ飛ぶ．さもなくば，$\boldsymbol{x}_0 = \boldsymbol{x}_{\text{new}}$ とし，Step 2 へ戻る．

(トンネリングのフェーズ)

Step 4：T をある値(初期温度)に設定する．

Step 5：(0, 1) の間の一様乱数を発生し $(-\pi/2, \pi/2)$ の数 P_i に変換し，(4)式により δx_i を発生する．これを変数の数だけ実行し，ベクトル $\boldsymbol{\delta x}$ を作る．

Step 6：(5)式により，新しい点 $\boldsymbol{x}$ を求める．新しい点が(6)式を満たせば，より深い谷の点がみつかったわけであるので，$\boldsymbol{x}_0 = \boldsymbol{x}$ として最小化のフェーズの Step 2 へ戻る．(6)式を満たさず，探索回数があらかじめ決められた回数(ITERS 回)に達していなければ，Step 5 へ戻る．

Step 7：あらかじめ決められた回数(ITERS 回)実行して(6)式を満たす点がみつからなければ T をスケジュールに従って少し小さくする(冷却する)．新しい T を使用して再度探索するため，Step 5 へ戻る．

Step 8：T の冷却スケジュールが終了しても(6)式を満たす点がみつからなければ，$\boldsymbol{x}^*$ を大域最適解とみなす．

最小化のフェーズには，説明の都合上，勾配法(最急降下法)が使用されている．ただし，簡単のため，ステップ幅 α は固定されている．勾配法の停止条件としては，(8.a)式，(8.b)式どちらを使用してもよい．一般的には，(8.b)式が使用されるが，本研究の数値例における計算では，全て(8.a)式を使用している．

ここで注意して欲しいのは，$\boldsymbol{x}^*$ は最小化のフェーズで求められた解であって，トンネリングのフェーズでは変化しないことである．$\boldsymbol{x}^*$からの増分がランダムに変化

し，新しい点 $\boldsymbol{x}$ が発生されることになる．

トンネリングのフェーズが実行されるごとに(6)式を満たす点がみつかれば，より深い谷に属する点がみつかるということであり，最小化のフェーズでその谷の極値がみつかる．順次みつけられる局所解は，それぞれ，直前にみつけられた局所解よりも改良された解であり，目的関数の値は順次改良されていく．つまり，$\boldsymbol{x}^{*i}(i=1,\cdots,p)$ を順次みつけられた局所解とすると，

$$f(\boldsymbol{x}^{*1})>f(\boldsymbol{x}^{*2})>\cdots>f(\boldsymbol{x}^{*p})$$

が成り立つ．最後に(6)式を満たす解がみつからなければ，より深い谷は存在しないとみなし，最後の局所解 $\boldsymbol{x}^{*p}$ を大域最適解 $\boldsymbol{x}^{opt}$ とする．

しかし，トンネリングのフェーズで，無限回探索を実行するわけではなく，ある程度の回数で打ち切るわけであるから，当然乱数の発生ぐあいによって，探索漏れが生ずる危険性が絶えず存在する．これをある程度防ぐため，マルチスタート方式を採用する．マルチスタート方式とは，良く知られているように，乱数の種を変え，異なる初期値から出発し，何度も同じ手順を繰り返すもので，たとえば，10回繰り返せば，10回分の大域解が得られる．ただし，この方式を採用しても，問題の規模にあわせて，探索漏れの危険性は常に存在する．

T の冷却スケジュールについてはFSAのように厳密に考える必要は全くない．FSAとは全く異なる概念に基づいているので，適切な徐々に小さくなるようなスケジュールを与えれば良い．ただ，本研究ではFSAをある程度は参考にして，たとえば1，1/2，1/3，1/4，…あるいは，1/2，1/4，1/6，…のように冷却することにしている．

さて，このアルゴリズムでは，もし谷の入口や途中に停止条件(8)式などを満たすような，非常になだらかな所があると，局所解に到達していなくてもトンネリングのフェーズに飛んでしまうということが起きる．しかし，このような場合でも，トンネリングのフェーズで同じ谷の改良点か，あるいはほかの谷の改良点をみつけることができ，最終的にはある谷の局所解にいたるので，これといった問題は起きない．

2.2 トンネリングフェーズの詳細について

上記アルゴリズムにおいて根本的な疑問は，どのようなメカニズムによりトンネル現象が起こるのであろうか，つまり，どのような理由で(6)式を満たすより深い谷に属する点がみつかるのであろうか，ということであろう．本セクションでは，この疑問に答えるために，トンネリングのフェーズの詳細について説明する．

最小化のフェーズで局所解に捕まったとする．この局所解を脱出するために，(4)式を使って増分を発生し，(5)式によってより深い谷の点をみつけるためには，決定変数の空間をできるだけ幅広く探索する必要が生ずる．$\boldsymbol{\delta x}$ を幅広く発生させるために，ここではCauchy分布を使用する．1次元のCauchy分布の密度関数は以下のように表わされる．

$$\frac{1}{\pi}\frac{T_c}{x^2+T_c^2} \tag{9}$$

ここで，(9)式からもわかるように，分布関数は平均が定義されず，その裾野は無限遠まで広がっており，これによって非常に広い領域を探索することができる．これはFSAと全く同じ理由でCauchy分布を使用することを意味している(ここで T_c は温度と呼ばれる)．

この累積分布関数を以下のようにおく．

$$F_c=F(x\leqq x_c)=\frac{1}{\pi}\int_{-\infty}^{x_c}\frac{T_c}{x^2+T_c^2}dx \tag{10}$$

$y=\mathrm{Arctan}\,z$ の y の $(-\pi/2,\pi/2)$ の主値を使って積分し x_c について整理すると，以下の式を得る[7]．

$$x_c=T_c\tan(\pi F_c-\pi/2)=T_c\tan P \tag{11}$$

ここで，$P=\pi F_c-\pi/2$．

x_c を δx_i，T_c を T，P を P_i とすると，(11)式は(4)式になる．各次元を独立とし，n 個の δx_i を発生すると，(5)式による探索が可能となる．FSAと同じように，Cauchy分布という無限に広がる裾野をもち，平均が定義されない分布関数に従ってランダムに増分を発生させ，幅広く探索することにより，より深い谷の点をみつけることが容易になる．

さらに，温度が高ければ(本研究ではFSAのように小さな係数 ρ を(4)あるいは(11)式の右辺に乗ずることはないので，何千度などという高温から出発することはないが)より遠くの点を探索する確率が高くなり，温度を徐々に低くしてゆくことによって，より近くの点を探索する確率が高くなる．このようにCauchy分布を使用することに加え，温度を調節することによって，非常に幅広い範囲の探索が可能になるわけである．

2.3 アルゴリズムに関する考察

本セクションでは，関係の深いFSAなどと関連づけて，本手法について異なる角度から検討を加えてみたい．

本手法において，最小化のフェーズでは(7)式で，トンネリングのフェーズでは(4)，(5)式で新しい点を探索する．これを一つの式にまとめると以下のように表わされる．

$$x_i=x_{0,i}-\eta\alpha\left(\frac{\partial f}{\partial \boldsymbol{x}}\right)_i+(1-\eta)T\tan P_i \tag{12}$$

これを，さらに以下のように書き表わす．

$$x_i=x_{0,i}-\eta\alpha\left(\frac{\partial f}{\partial \boldsymbol{x}}\right)_i+(1-\eta)\rho T_a\tan P_i \tag{13}$$

ここで，$T=\rho T_a$．ρ は非常に小さな正数，i はベクトル

の要素を示す($i=1, \cdots, n$).

(12), (13)式において, η は 0 または 1 の値をとる係数で, 最小化のフェーズでは $\eta=1$ に, トンネリングのフェーズでは $\eta=0$ になっていると考えることができる(ただし, トンネリングのフェーズでは $\boldsymbol{x}_0$ は局所解 $\boldsymbol{x}^*$ に固定される). $\eta=0$ の場合, $\boldsymbol{x}_0$ が $\boldsymbol{x}^*$ に固定される点を考慮しなければ, (13)式は FSA における探索式であり, (13)式全体としては, FSA の式に勾配法による項を付け加えた式になっている. η を 0, 1 以外の適当な値に設定して, (13)式全体を使って, アニーリングを行うということも可能であろう. 実際, 文献 7) (pp. 89〜90)では, バックプロパゲーションの重み改良の式に, 上式のように Cauchy 探索の式を η を使って結合し, 良い学習結果を得たことが報告されている. バックプロパゲーションでうまく行くものであれば, η の値を適切な値に固定するか, またはアダプティブに変更して, (13)式を使用してアニーリングを行う方法により, 最適化問題を解くことも可能かもしれない.

しかし, 本稿ではあくまでもトンネリングアルゴリズムの概念の応用を考えている. $\boldsymbol{x}_0$ が $\boldsymbol{x}^*$ に固定されるかどうかが, 式の上でのトンネリングのフェーズと FSA との違いであり, FSA と非常に関係深い式を使うが, 本手法は, $\eta=1$ でまず局所解 $\boldsymbol{x}^*$ を探索し, 局所解がみつかったところで, $\eta=0$ に切り替えて, $\boldsymbol{x}_0$ を $\boldsymbol{x}^*$ に固定し, 温度を調整しながらより深い谷の点をみつけるという, FSA とは全く異なる概念に基づいている. また, FSA のように山登りをある確率で許すことも全くない. ただ, FSA とは Cauchy 分布の使用と温度の調整で, できるだけ広範囲な探索を行い, 局所解を脱出しようという, 探索に対する姿勢が同じであるといえよう.

さて, 幅広く探索するためのキーポイントの一つが Cauchy 分布に基づく(4)式の使用であった. 特に, タンジェント関数が鍵となるわけであるが, タンジェント関数と同じような形をもつ関数を使用しても, ある程度うまくいくことが予想される. たとえば, つぎのような関数で δx_i を発生させることを考える.

$$\delta x_i = T \ln \frac{1+P_1}{1-P_1}$$

この式では, P_1 は $(-1, 1)$ の間の数.

この式はある確率分布から導かれた式ではなく, ニューラルネットワークに使用されるシグモイド関数 $P_1(x)=(1-e^{-x/T})/(1+e^{-x/T})$ を単に x について整理して導かれたものである. 経験的にいって, この関数を使用してもうまくアルゴリズムは働くが, しかし, タンジェント関数を使用したときよりも, 大域解を逃す確率が増える.

2.4 簡単な制約のある問題について

先のアルゴリズムでは, 無制約の最適化問題を考えた. つぎに, 制約領域が n 次元超立方体,

$$X=\{\boldsymbol{x} \mid a_i \leqq x_i \leqq b_i \quad (i=1, 2, \cdots, n)\}$$

の場合を考える. 一般的に, 制約のある問題は, ペナルティ関数法などを利用して解けば良いのだが, ここでは超立方体のような簡単な制約領域を考えているので, 以下のような簡易な方法を取ることにする.

制約領域が上に示す超立方体 X のような場合, Step 1 において, 初期値は X に属するように発生させる. Step 3 と Step 6 を以下のように変更する.

Step 3': もし $|\partial f/\partial x_i|_{\boldsymbol{x}=\boldsymbol{x}_{\text{new}}}|<\delta$ ($i=1, 2, \cdots, n$), あるいは $\|\partial f/\partial \boldsymbol{x}|_{\boldsymbol{x}=\boldsymbol{x}_{\text{new}}}\|<\omega$ であれば, 局所解に収束したとし, $\boldsymbol{x}^*=\boldsymbol{x}_{\text{new}}$ とし, この解を記憶し, トンネリングフェーズへ飛ぶ. もし $\boldsymbol{x}_{\text{new}}$ が超立方体 X を飛び出せば, つまり, $\boldsymbol{x}_{\text{new}} \notin X$ であれば, $\boldsymbol{x}_0 (\in X)$ を解として記憶し, 同じくトンネリングフェーズへ飛ぶ. さもなくば, $\boldsymbol{x}_0=\boldsymbol{x}_{\text{new}}$ とし, Step 2 へ戻る.

Step 6': (5)式により新しい点 $\boldsymbol{x}$ を求める. $\boldsymbol{x} \notin X$ であれば, この点 $\boldsymbol{x}$ を捨て, Step 5 へ戻り, 新しく $\boldsymbol{\delta x}$ を発生させる.
$\boldsymbol{x} \in X$ であり, 新しい点が(6)式を満たせば, より深い谷の点が制約領域内にみつかったわけであるので, $\boldsymbol{x}_0=\boldsymbol{x}$ として最小化のフェーズの Step 2 へ戻る. (6)式を満たさなければ, Step 5 へ戻る.

Step 3' において, $\boldsymbol{x}_{\text{new}}$ が最初に領域外へ飛び出した点であれば, $\boldsymbol{x}_0$ は領域内にあり, 多少の誤差はあるが, この点を境界上にある点とみなし, この点を最小化のフェーズの解とするということである. 誤差の問題は, 精度の高い方法を取ることにより解消されると思われる. ただし, 上記の簡易な方法では, 超立方体の境界や領域内に, 最急降下方向ではないが目的関数を改良する方向がある場合でも, トンネリングのフェーズへ飛んでしまうという欠点があることに注意されたい.

$g_i(\boldsymbol{x}) \leqq 0$ $(i=1, \cdots, N)$ などのような制約条件があるときも, もし制約式が簡単なものであり, 個数も少なければ同様に扱うことが可能であろう. しかし,

$$g_i(\boldsymbol{x}) \leqq 0 \ (i=1, \cdots, N), \quad h_j(\boldsymbol{x})=0 \ (j=1, \cdots, M)$$

などのような制約式が存在する一般的な場合は, 上述したように, ペナルティー関数法などを適用するのが良いと思われる. しかし, それに伴って計算時間の問題など, いろいろな困難も予想されるので, この問題については, 今後の課題とし, 稿を改めて報告したいと思う.

3. 数値例について

このセクションでは，本稿で提案したアルゴリズムを用いて例題を解くことにする．

〔例題1〕　以下の問題を考える[8]．

$$\min f(\boldsymbol{x})=\frac{1}{2}\sum_{i=1}^{n}(x_i^4-16x_i^2+5x_i) \qquad (14)$$

この目的関数は少なくとも 2^n 個の局所解をもち[8]，その内の一つ

$\boldsymbol{x}=(-2.90353, -2.90353, \cdots, -2.90353)$

が大域最小解になる．

Case 1：$n=2$ の場合．

この場合，局所解は4個あり，その内の一つが大域解 $(-2.90353, -2.90353)$ である．また，$f^{\text{opt}}=-78.3323$ である．初期値を $-10\leqq x_i\leqq 10\ (i=1, 2)$ の範囲に発生させる．$\alpha=0.001$，$\delta=0.001$ とする((8.a)式を勾配法の停止条件に使用する)．

温度の冷却スケジュールは $T=1/4, 1/6, 1/8, 1/10$ とし一つの温度 T(たとえば1/4)あたりの新しい点 $\boldsymbol{x}$ の探索回数 ITERS は500回とする．マルチスタートの回数は10回とし，毎回乱数の種を変え，10個の大域解とみなされ得る点を求める．上のようにパラメータを設定すると，マルチスタートの回数10回中ほぼ全て，大域最小解に収束する．ランダム現象なので求められた解を表示するのはさほど意味がないので，解の一つを示すと $(-2.90351, -2.90354)$ であり，$f^{\text{opt}}=-78.3323$ である(計算は Basic を用い，単精度で行った)．

この例では，上のようにパラメータを設定すれば，大域最適解を逃すことは全くない．ただ，あくまでもランダム現象なので，乱数の種によっては局所解に収束しないとはいい切れないが，10回のマルチスタート中，大域解を逃がす確率は非常に小さいと思われる．なお，アルゴリズムからいって当然のことであるが，大域解を得ることにはさほど計算時間を要しないが，大域解であることを確認するため(より深い谷がないことを確かめるため)に非常に多くの計算時間を必要とする．

Case 2：$n=10$ の場合

この場合，局所解は $2^{10}=1024$ あり，$f^{\text{opt}}=-391.662$ である．10次元の問題なので以下のようにパラメータを設定する．

$\alpha=0.002$，$\delta=0.001$，ITERS$=2000$

T のスケジュールは $T=1/4, 1/6, 1/8, 1/10$

初期値は $-10\leqq x_i\leqq 10\ (i=1, \cdots, 10)$ の範囲に発生させる．マルチスタートの回数は10回．

ここで，注目しなければならないのは ITERS の回数で，Case 1 の4倍になっている．これは，変数の次元が2から10になり，探索すべき範囲が非常に広くなったことからも当然のことで，この ITERS が小さいと大域解を逃がす率が大きくなる．また，大きくなりすぎると大域解をみつけるには良いのだが，計算時間がかかりすぎるようになり，T の冷却スケジュールと共に，本手法の非常にクリティカルな要素となっている．

さて，上のようにパラメータを設定し，何度か計算を行ったが，ほとんどの場合10回のマルチスタートのうち，10回大域解が求められている．最も悪い場合で10回中1度だけ $(-2.9035, -2.9035, 2.7468, -2.9035, -2.9036, -2.9035, -2.9035, -2.9035, -2.9035, -2.9035)$，$f^*=-377.525$ に収束し，大域解を逃したことがあるが，それ以外は全く逃がすことはなかった．大域解を逃したのは，変数の次元が上がり探索領域が広がったためであろう．変数の次元の高い問題ほど ITERS の値を大きくしなければならない．解の一つは $(-2.9035, -2.9035, -2.9035, -2.9035, -2.9036, -2.9035, -2.9035, -2.9035, -2.9035, -2.9035)$ で，$f^{\text{opt}}=-391.662$ である．ただ，先に述べたように，あくまでもランダム現象なので，乱数の種によっては局所解に収束しないとはいい切れないが，パラメータさえ適切に設定すれば，10回中大域解を逃がす確率は非常に低いと思われる．

〔例題2〕　2次元の Shubert 関数

$$\min f(x_1, x_2)=\{\sum_{i=1}^{5} i\cos[(i+1)x_1+i]\}*\{\sum_{i=1}^{5} i\cos[(i+1)x_2+i]\} \qquad (15)$$

$$\text{subject to}\quad -10\leqq x_i\leqq 10\ (i=1, 2) \qquad (16)$$

(＊は乗算を表わす)

今度は(16)式のような制約があるので，セクション2.4節で考えたアルゴリズムを使用する．

この目的関数には760の局所解があり，その内の18個が大域最適解である[8]．大域最適解は参考文献を見ていただければ良いのだが，求められた解と比べていただくために以下の **Table 1** に表示する．なお，$f^{\text{opt}}=-181.731$ である．

今度はパラメータを以下のように設定する．

$\alpha=0.0001$，$\delta=0.005$，ITERS$=1000$，初期値は当然，領域(16)式内に発生し，マルチスタートの回数は20回とする．$T=1/4, 1/6, 1/8, 1/10$ である．

求められた解を **Table 2** に示す．全ての場合に大域最適解が得られている．乱数の種が異なれば，当然 Table 2 とは異なる解が得られる．また，局所解に捕まる場合もあるかもしれないが，上の結果から，非常に高い確率で大域最適解が得られると思われる．

〔例題3〕　Six-hump camelback 関数

$$\min f(x_1, x_2)=[4-2.1x_1^2+x_1^4/3]x_1^2+x_1x_2 +[-4+4x_2^2]x_2^2 \quad (17)$$

$$\text{subject to } -3\leqq x_1\leqq 3,\ -2\leqq x_2\leqq 2 \quad (18)$$

この問題の局所解は 6 点存在し[8]，その内の 2 点 (−0.0898, 0.7126)，(0.0898, −0.7126) が大域最適解であり，$f^{opt}=-1.03163$ である．上記問題は以下のようにパラメータを設定すると，大域最適解をほぼ逃すことなくみつけることができる．

$\alpha=0.001$，$\delta=0.001$，ITERS=1000，マルチスタートの回数 =10，$T=1/4, 1/6, 1/8, 1/10$ である．

二つの大域最適解は，ランダム現象なので，偏って得られることもあるが，何度か繰り返すと，それぞれがだいたい 1/2 の確率で得られる．

この問題を (18) 式ではなく，以下の (19) 式を制約式として解く．

$$-3\leqq x_1\leqq 3,\ -0.7\leqq x_2\leqq 2 \quad (19)$$

(0.0898, −0.7126) は制約式の領域外の点となる．この問題を前述したパラメータ値を使い，マルチスタートの回数 5 回で解くと，**Table 3** に示す解が得られる．ほぼ境界上にある点 (0.09324, −0.69992)，$f^*=-1.03023$ に捕まると，より深い谷へは脱出できない．しかし，目的関数値の大域最適解との差は，0.0014 である．これは，この点に捕まると，探索すべきより深い谷の領域が非常に小さく，大海に失った宝石を探すようなもので，より深い谷に属する点をみつけることが非常に困難なことを示している．一般的にいって，本手法では，局所解よりもより深い谷の，局所解と水平レベルにある，谷の間口が非常に小さいような問題では，局所解からの脱出はかなり難しいといえよう．これを一次元で図示したものを **Fig. 2** に示す．たとえば，Fig. 2(a) のような場合，Δ_1 は広いので，局所解 A からの脱出は容易であろう．しかし，Fig. 2(b) のような場合，もし局所解 B に捕まると Δ_2 が非常に狭いので，B からの脱出には困難が予想される．このような場合でも，マルチスタート方式によって，次回に C が得られれば，Δ_3 はかなり広いので，大域解の属する谷がみつかる確率が高くなる．このように，マルチスタート方式により，本例題でも 5 回中 4 回大域最適解に収束しており，ここにマルチスタート方式の意義があるといえよう．また，本手法は探索方式から判断して，探索すべき決定変数の空間が非常に広く，谷と谷の間隔が非常に離れているような問題には適していないと思われるが，これもマルチスタート方式によりある程度はカバーできるのではないかと思われる．

Table 1 Global minimums in Example 2

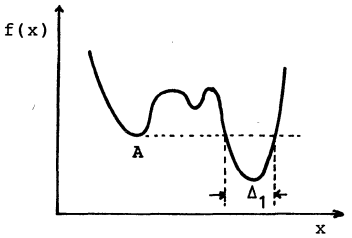
x 1	x 2
5.4828	−1.42513
−0.80032	4.85805
4.85805	−0.80032
−7.0835	−7.70831
−0.80032	−7.70381
5.48286	−7.70831
−7.70831	−7.0835
−1.42513	−7.0835
4.85805	−7.0835
−7.0835	−1.42513
−0.80032	−1.42513
−7.70831	−0.80032
−1.42513	−0.80032
−7.0835	4.85805
5.48286	4.85805
−7.70831	5.48286
−1.42513	5.48286
4.85805	5.48286

Table 2 Obtained solutions by the proposed method in Example 2

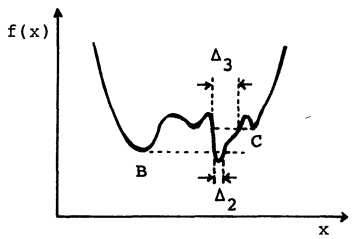
Start _ No.	x 1	x 2	fopt
1	−7.08531	4.85806	−186.731
2	−7.70831	−0.80032	−186.731
3	−7.08351	−7.70831	−186.731
4	−1.42513	5.48286	−186.731
5	−0.80032	−1.42513	−186.731
6	4.85806	−0.80032	−186.731
7	5.48286	4.85806	−186.731
8	−0.80032	4.85806	−186.731
9	−0.80032	4.85806	−186.731
10	−7.08351	4.85806	−186.731
11	−1.42513	−7.08351	−186.731
12	−7.08351	−1.42513	−186.731
13	4.85806	−7.08351	−186.731
14	−7.08351	−7.70831	−186.731
15	−0.80032	−7.70831	−186.731
16	4.85806	5.48286	−186.731
17	−1.42513	5.48386	−186.731
18	−1.42513	5.48386	−186.731
19	−1.42513	5.48286	−186.731
20	−1.42513	−7.08351	−186.731

Table 3 Obtained solutions by the proposed method in Example 3

Start _ No.	x 1	x 2	fopt
1	0.09324	−0.69992	−1.03023
2	−0.08971	0.71264	−1.03163
3	−0.08997	0.71267	−1.03163
4	−0.08997	0.71267	−1.03163
5	−0.08997	0.71269	−1.03163



（a） The case where proposed algorithm works well.



（b） The case where proposed algorithm may not work so well.

Fig. 2

そのほか，テスト関数としては，Three-hump camel-back 関数[3]，Hosaki 関数[3]などを使用したが，いずれの場合も大域最適解がほぼ逃すことなく求められており，非常に良い結果が得られている．

以上，計算は全て Basic で単精度で行った．プログラムは注釈行を含めても 70 行程度のもので，非常に簡単にプログラムが組めるのも本手法の大きな特徴である．例題 1 の Case 1 の場合のプログラムが文献 4）にあるので参考にしていただきたい．ただ，本稿と同じ例題を解く場合でも，計算を倍精度で行う場合などは，α や δ などのパラメータ値を変更する必要があると思われる．また，異なる問題に対しては，その問題にとって適切であるパラメータ値を設定しなければならない．

4. 他手法との比較について

本稿の目的は，局所解を脱出するために広く探索が可能な Cauchy 分布を使用し，徐冷という過程をとおしてシステマティックに探索を行う手法の提案である．ここでは簡単に FSA との比較を中心にその有効性を考えてみたい．

まず，数値例 1 の 2 次元問題を参考にして FSA と比較したい．FSA を 10 回マルチスタートさせた結果のなかから収束の早い，典型的な例を **Fig. 3** と **Fig. 4** に示す（Fig.3 には 2500 ステップまで Fig. 4 には 3000～5000 ステップが表示されている）．横軸は目的関数値の評価回数である（ただし，収束を早めるために初期温度は $T_0=100$ 度，一つの温度での探索回数は 100 回，$x_i=x_{0,i}+\rho T\tan P_i$ $(i=1,2)$ で探索し，$f(\boldsymbol{x})\geqq f(\boldsymbol{x}_0)$ の場合は $(0,1)$ の間の一様乱数を発生させ，その乱数が $1/(1+\exp(\Delta f/T))$ より小さいときアクセプトする．$\rho=0.01$，T は $1/(t+1)$ に比例して冷却する．t は時間：詳細は文献 6）を参照されたい．なお，FSA の概略を参考のため付録の項に示す）．また，数値例と同じ条件で，本手法を用い 10 回マルチスタートさせた結果のなかから収束の最も速いものと遅いものを同じく Fig. 3 に示す．ここでは，勾配

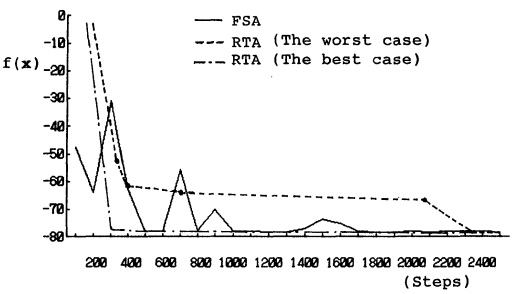


Fig. 3 FSA and Random Tunneling Algorithm (RTA)

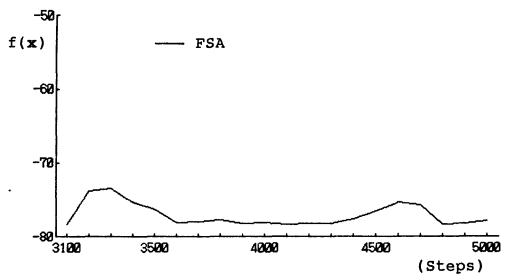


Fig. 4 FSA

法の 1 step を 1 回の目的関数の評価としている．これを比較すると Fig. 4 に見るように FSA では大域解に近づいても落ち着かず，近傍を動き回る．これは，FSA では大域解に落ち着くのに非常に時間がかかることを表わしている．また，この条件設定では大域解に収束したのは 10 回中 7 回であり，本稿で提案している手法が大域解をほとんど逃すことがないことに比べかなり悪い．これは上記のパラメータ設定のなかに不適切なものがあったせいかもしれない．しかし，もし初期温度を高くし，一つの温度当りの探索回数を増やせば，この結果は改善されるとしても，大域最適解に落ち着くまでにさらに長時間がかかるということである．総じて，シミュレーテッド・アニーリングでは，冷却しつくすまでに非常に時間がか

かるという弱点がある．本稿で提案している手法でも，大域解とみなされる点よりもより深い谷が存在しないことを確認するため，上の数値例において2000ステップ程度さらに必要であるが，山登りを許すことはないので，近傍を動き回るなどということは全く起こらず，大域解に落ち着くのに FSA のような長時間を必要とはしない．

さて，Cauchy 分布を使用した本手法では，温度 T が低いと局所解の近傍が探索される確率が高い．これはニューラルネットワークの学習問題などの大域最適化の応用問題でも，局所解の近傍により深い谷が存在することが多く，現実的な探索である．しかし，Cauchy 分布を使用することにより，ロングジャンプが可能であり，それに加え温度のシステマティックな変更により，局所解から離れた領域の探索も可能であるため，たとえば正規分布に従った探索よりも，より広い領域を探索できるといえよう．また，本手法において，問題に適切な温度が一つ設定できれば，ほかの温度をトライする時間が節約でき，より有効ではあるが，実際には一つの適切な温度を求めることは難しく，徐冷をとおしてシステマティックに変更したほうがより現実的であるといえよう．

最後に，本手法では徐冷型の探索を行っているが，FSA のように収束に関係しているわけではないので，温度係数を徐々に大きくする加熱型を使用してもよい．加熱型を使用すると目的関数の改良幅の小さな点がトンネリングのフェーズの結果得られることが多い．大幅な改良ができないのでトンネリングのフェーズへ行く回数も増えるが，最終的には解が同じように求められるので，現在の段階ではどちらが有利ともいえない．これも今後の課題としたい．

以上，簡単な比較を行ったが，最適化の諸手法は，それぞれに長所と欠点があり，プログラムを作る段階，パラメータの設定の段階などを含め，いろいろな観点から比較されるべきもので，簡単に結論を出すべきものではないであろう．ここでは簡単に FSA などと比較したが，いろいろな観点からの厳密な比較は今後の課題としたい．

5. おわりに

本稿では，トンネリングアルゴリズムの概念を応用した手法について提案した．提案した手法の大きな特色はトンネリングのフェーズに徐冷型のランダム探索を取り入れたことにある．その特徴は：

1) 最小化のフェーズで求められた局所解からの増分を，設定した温度のもとで，無限に広がる裾野をもつ*(平均が定義されない)，Cauchy 分布に従って発生する，

2) 上記 1)に加え，温度の冷却スケジュールを調節することにより，広く決定変数の空間を探索し，局所最適解から脱出する，

などである．

本稿では，微分可能関数の大域最適化のみを考えたが，決定変数が連続な実数である問題であれば，目的関数が微分不能でも不連続関数でも，最小化のフェーズにランダムサーチを用いれば[1]，同様に解くことができる．さらに，Cauchy 分布を用いたランダムサーチを使用しても良いが，それに伴っていろいろな問題が起こる可能性もあり，これは今後の課題とし，稿を改めて紹介したい．

本手法の非常にクリティカルな部分は，1)パラメータの値の設定，特に一つの温度に対する探索回数 ITERS の値，と 2)温度 T の冷却スケジュール，であろう．この二つが適切でないと大域最適解に収束するのは難しいと思える．T に対しては 1/4, 1/6, 1/8, 1/10 のように例題では冷却したが，決定変数の次元の高い問題では，より細かに冷却する必要があろう．同様に ITERS の値についても，例題では 500～2000 程度の値を用いたが，これも変数の次元の高い問題になるほど大きくしなければならないであろう．

本手法はまだ見つけられたばかりであり，パラメータ値の設定など，たくさんの課題が残されている．しかし，数値例のところで検討したような問題に対しては非常に有効であり，ほぼ逃すことなく大域最適解が得られている．ほかのテスト関数でも，同様に，非常によい結果が得られていることからも，今後いろいろな問題への応用に対しても，大変期待できる手法であると思われる．現在，本稿の手法を一般化したアルゴリズムで，数十変数のニューラルネットワークによる学習問題や適応制御問題が解かれており，これについては稿を改めて報告したい．

参考文献

1) F. Archetti and F. Schoen: A Survey on the Global Optimization Problem, General Theory and Computational Approaches, Annals of Operations Research, **1**, 87/110 (1984)

2) A. V. Levy and A. Montalvo: The Tunneling Algorithm for the Global Minimization of Functions, SIAM J. Sci. Stat. Comput., **6**-1, 15/29 (1985)

3) L. Pronzato, E. Walter, A. Venot and J. F. Lebruchec: A General-Purpose Global Optimizer: Implementation and Applications, Mathematics and Computers Simulation, **26**, 412/422 (1984)

4) 島　孝司：徐冷型ランダムトンネリング手法による大域的最適化について，第 18 回システムシンポジウム，第 16 回知能システムシンポジウム，第 2 回ニューラルネットワークシンポジウム，合同シンポジウム講演論文集，387/392 (1992)

5) 須貝，平田：組合せ最適化アルゴリズムとその応用，計測と制御，**29**-12, 1084/1091 (1990)
6) H. Szu and R. Hartley: Fast Simulated Annealing, Physics Letters A, **122**-3, 4, 157/162 (1987)
7) P. D. Wasserman: Neural Computing, Theory and Practice, Van Nostrand Reinhold, N. Y. (1989)
8) Y. Yao: Dynamic Tunneling Algorithm for Global Optimization, IEEE Transactions on Systems, Man, and Cybernetics, **SMC**-**19**-5, 1222/1230 (1989)

《付　　録》

ファースト・シミュレーテッド・アニーリング法(FSA)はシミュレーテッド・アニーリング法(SA)に比べ馴染みが薄いので，アルゴリズムの概要をここで紹介することにする．以下では，T を温度とし，$T_0, T_1, \cdots, T_t, \cdots, T_q$ のように冷却する．ここで，t は冷却スケジュールを示すカウンタで，0, 1, 2, … のように最終値 q まで変化する．また，一つの温度当りの探索回数を M とする．

Step 1：$t=0$ とし，初期温度 T_0 を設定する．初期値 $x_0 \in X$ をランダムに発生する．

Step 2：$T_t=T_0/(1+t)$ とし，$m=1$ に設定する．

Step 3：$x_i=x_{0,i}+\rho T_t \tan P_i\ (i=1, \cdots, n)$ により新しい点 $\boldsymbol{x}$ を発生する．

a．$f(\boldsymbol{x})<f(\boldsymbol{x}_0)$ ならば $\boldsymbol{x}_0=\boldsymbol{x}$

b．$f(\boldsymbol{x})\geqq f(\boldsymbol{x}_0)$ ならば

b.1. $\mathrm{random}(0, 1)<1/(1+\exp((f(\boldsymbol{x})-f(\boldsymbol{x}_0))/T_t))$ のとき $\boldsymbol{x}_0=\boldsymbol{x}$

b.2. そのほかのとき $\boldsymbol{x}_0$ は更新しない．

Step 4：$m=m+1$ とし $m\leqq M$ ならば Step 3 へ戻る．

Step 5：$t=t+1$ とし $t\leqq q$ ならば Step 2 へ戻る．

$t>q$ ならば終了．

ここで，random(0, 1) は (0, 1) の間の値をとる一様乱数を示す．P_i は (0, 1) 間の一様乱数を $(-\pi/2, \pi/2)$ 間の数に変換したものである．ρ は小さな正数である．

[著　者　紹　介]

島　　孝　司（正会員）

1982年，Case Western Reserve 大学システム工学科博士課程修了，現在，金沢女子短期大学情報処理学科教授．大規模システム，大域最適化，ニューラルネットワークに関する研究に従事．システム制御情報学会，IEEE などの会員(Ph. D)．