Proposal of adaptive range particle swarm optimization

領域適応型Particle Swarm Optimizationの提案

北山哲士[*1]　荒川雅生[*2]　山崎光悦

Proposal of Adaptive Range Particle Swarm Optimization

Satoshi Kitayama, Masao Arakawa, Koetsu Yamazaki
Department of Human & Mechanical Systems Engineering, Kanazawa University,
Kakuma-machi, Kanazawa-shi, Ishikawa, 920-1192, Japan

This paper proposes a new method which is called as the Adaptive Range Particle Swarm Optimization (ARPSO), based on the Adaptive Range Genetic Algorithm. That is, the active search domain is determined by using the mean and standard deviation of each design variable. In general, multi-points methods are utilized in the field of evolutionary computation. At the initial search stage it is preferable to explore the search domain widely, and is also preferable to explore the smaller search domain as the search goes on. To achieve this objective, new parameter which determines the active search domain is introduced. This new parameter gradually increases as the search goes on. Finally it is possible to shrink the search domain. The way to determine the maximum value of this new parameter is also shown in this paper. The optimum solution with high accuracy and a little number of function calls is obtained by prpoposed method in compared with the original Particle Swarm Optimization. Through numerical examples, the effectiveness and validity of proposed method are examined.

Key Words : **Global Optimization, Adaptive Range Particle Swarm Optimization, Optimum Design, Systems Engineering, Engineering Optimization**

1　緒言

多峰性関数の大域的最適解を求める方法は，しばしば確定的な方法と確率的な方法に分類されることが多いが，一方で探索点の数によって単点探索法と多点探索法に分類することも可能である．特に実行可能領域内に多数の探索点を配置し，多点同時探索により多峰性関数の大域的最適解もしくはそれに相当するような準最適解を求める方法は，大きく，関数の感度（勾配）を利用する方法と，関数の感度を利用しない方法に分類することができる．特に関数の感度を利用しない方法の中でも遺伝的アルゴリズム（GA）は，種々の問題への柔軟な適用が可能であることから多くの研究者に受け入れられ，様々な研究が行われている．しかし基本的にGAは設計変数を0 - 1のビットで表現するため，特に離散変数に対して有効である．

一方で関数の感度を利用せずに，連続変数から成る多峰性関数の大域的最適解もしくはそれに相当するような準最適解を求める方法として，Particle Swarm Optimization(PSO)が提案された[(1)]．PSOは，はじめに設定された側面制約条件内にランダムに探索点が配置される．探索点は位置と速度を持ち，それらを更新することにより，大域的最適解を求める方法である．PSOは連続変数の無制約最適化問題に対して，高い精度で大域的最適解を求めることができ，筆者らはPSOを連続変数と離散変数から成る混合変数問題への適用を行い，その有効性を示した[(2)]．

しかしながら，PSOも他の多点探索法と同様に，はじめに設定した側面制約条件は固定され，探索が十分進んだ段階でもその側面制約条件の変更はない．このため，探索が十分進んだ段階でも，ある意味で無駄な探索領域が設定されたまま探索を行うことになる．多点同時探索で大域的最適解を求める進化的計算法と称される多くの方法では，全探索点が目的関数を最もよくする探索点へ向うため[(3)]，探索点から得られる何らかの情報を利用した有効な探索領域の絞込みは必要であると考えられる．また有制約最適化問題に適用する際，制約条件をペナルティ関数として扱う方法が一般的であるが，ペナルティ関数に対するペナルティ係数は経験的に決められていることが多い[(2)]．

ここで関数の感度を利用しない多点探索法の課題をまとめると，次のようになると思われる．

(P1) 探索点のばらつきを考慮した探索領域の設定．

＊ 原稿受付　平成？？年？？月？？日
＊1正員，金沢大学工学部（〒920-1192　金沢市角間町）．
＊2正員，香川大学工学部（〒761-0396　高松市林町2217-20）．
E-mail: kitagon@t.kanazawa-u.ac.jp

(P2) 探索状況に応じた探索領域の変更.
(P3) 探索回数を考慮した有効な探索領域の絞込み.
すなわち探索点の集中化による解の精度向上.
(P4) 制約条件の取扱い.

PSOは，筆者らが既に示したように[2]，関数の感度を利用しないにもかかわらず，その構造上，勾配法との類似性があり，確定的な方法としての要素が極めて強いと思われる．本論文では特に上記（P1）～（P3）に焦点を当て，確定的な要素の強いPSOに統計的な要素を含ませることにより，有効な探索領域の設定と探索点の集中化によって，最適解の精度向上を狙う．PSOが感度を利用しない大域的最適化法の一つであることを考えれば，探索領域の絞込みにより，精度の高い大域的最適解を得ることが目的であるが，仮に大域的最適解が得られなくとも，精度の高い次善の解を得ることも重要であると考えられる．（PSOに確率的な要素を含ませ，多峰性関数の局所的および大域的最適解を求める方法が提案されているものの[4]，探索領域の絞込みや探索点の集中化を狙うものではないため，本論文で着目した点とは大きく異なる．）また(P4)の対策として，探索点の情報を利用して制約条件をペナルティ関数として扱った際のペナルティ係数を自動的に決めることも可能であると考えた.

そこで本論文では，筆者らの一人が開発した領域適応型遺伝的アルゴリズム[5]（ARGA)を参考に，各設計変数の平均・標準偏差を利用して，探索状況に応じて探索領域が適宜変更される領域適応型Particle Swarm Optimization(Adaptive Range Particle Swarm Optimization: ARPSOと略記）を提案する．いくつかの数値計算例を通じてその有効性を検討する.

2 Particle Swarm Optimization

PSOは，鳥や魚などの群れの行動，さらには人間の社会活動というものは，集団を構成する個々の情報を共有しながら進化を続けているということを基にした最適化手法である．PSOは，探索点（Particle)が持つ最良の情報（p-best)と，探索点から形成されるグループ（Swarm)の最適値（g-best)から，過去の探索履歴を考慮して連続変数の多峰性関数の大域的最適解を求める手法であり，集団はg-bestへ向い探索を行う．PSOでは各探索点が「位置」と「速度」を持ち，集団で探索を行い，各探索点の位置と速度を更新しながら，最適解を探索する方法である.

2.1 位置と速度の更新 k 回目の探索において，探索点 d の位置 x_d^k と速度 v_d^k を用いて，$k+1$ 回目の位置 $\mathbf{x}_d^{k+1}$ と速度 v_d^{k+1} は，次の式を用いて更新される.

$$x_d^{k+1} = x_d^k + v_d^{k+1} \quad (1)$$

$$v_d^{k+1} = w\, v_d^k + c_1 r_1 (p_d^k - x_d^k) + c_2 r_2 (p_g^k - x_d^k) \quad (2)$$

式（2）において，r_1 と r_2 は[0,1]の乱数である．また c_1 と c_2 はパラメータであり，一般には

$$c_1 + c_2 \le 4 \quad (3)$$

となるように，c_1 と c_2 は決められている．また w は慣性項と呼ばれるパラメータであり，線形的に減少する．p_d^k は，探索点 d が k 回目までの探索において，今までで訪れた最良の解(p-best)を表す．一方，p_g^k は k 回目の探索における群れ全体の中での最良の解(g-best)を表す．式（1），（2）を用いて探索点を更新するモデルは通常，g-bestモデルと呼ばれる．また式（2）中の p_g^k を，k 回目までの探索で目的関数値を最良にした探索点 p_g で置き換えたモデルは，最良値保存型モデルと呼ばれる.

2.2 基本アルゴリズム PSO（g-bestモデル）の基本アルゴリズムは次のようになる.

(STEP1) 探索点数，最大探索回数を決める．また側面制約条件を設定する.
(STEP2)各探索点に対して，ランダムに初期位置 x_d^k と初期速度 v_d^k を決める．$k=1$ とする.
(STEP3)各探索点に対して，関数値を計算する.
(STEP4) p_d^k と p_g^k を求める.
(STEP5)各探索点の速度と位置を式（1），(2）に従い更新する．また慣性項 w を以下の式に従い更新.

$$w = w_{\max} - (w_{\max} - w_{\min}) / k_{\max} \times k \quad (4)$$

(STEP6)探索回数 k が最大探索回数以下なら $k = k+1$ としてSTEP3へ戻る．そうでなければ，探索終了.

PSOの近傍や勾配法との類似については，文献（2）を参照されたい．上記アルゴリズムからも判るように，探索領域自体（すなわちはじめに与えられた側面制約条件）は変化することはない．慣性項 w は探索回数に応じて線形的に減少するが，これは既に筆者らが別報[2]で示したように，探索点の近傍自体を小さくするための要素であり，探索領域自体を変化させる要素ではない．さらにPSOでは，探索が進むにつれ，探索点は集中化することが報告されているが[3]，探索領域の絞込みはできない.

3 提案する方法

3.1 問題設定 本論文では，側面制約条件と挙動制約条件の下において単一の目的関数を最小化する問題を対象とする.

$$Find \quad \boldsymbol{x} = (x_1, x_2, \cdots, x_{ndv})^T \quad (5)$$

$$Such \quad as$$

$$f(\boldsymbol{x}) \to \min \quad (6)$$

$$Subject\ \ to$$

$$x_i^L \le x_i \le x_i^U \quad i=1,2,\cdots,ndv \qquad (7)$$

$$g_j(x) \le 0 \quad j=1,2,\cdots,ncon \qquad (8)$$

上式において，x は設計変数ベクトル，ndv は設計変数の数を表す．$f(x)$ は連続変数から成る最小化すべき目的関数であり，x_i^L と x_i^U は i 番目の設計変数に直接課せられる側面制約条件の下限値と上限値である．また $g_j(x)$ は挙動制約条件であり，$ncon$ は挙動制約条件の数を表す．以下では，挙動制約条件を無視し，側面制約条件下で単一の目的関数を最小化する問題を対象として述べる．挙動制約条件がある場合は3．7 節で述べる．

3．2　探索領域の設定　初期探索（$k=1$）のときは，直前の探索における情報がないため，2．2 節で述べたPSO を適用する．これにより i 番目の設計変数の平均 m_i と標準偏差 s_i が求まる．探索領域の設定は式（9）で表される正規分布を基調とし，この正規分布縦軸の値を利用して，式（10）のように探索領域を次のように設定する．

$$N(x)=\exp\left(-\frac{(x_i-m_i)^2}{2s_i^2}\right) \qquad (9)$$

$$m_i-\sqrt{-2s_i^{L2}\log a} \le x_i \le m_i+\sqrt{-2s_i^{R2}\log a} \qquad (10)$$

式（10）中の s_i^L と s_i^R は左右別々に与えられる i 番目の設計変数の標準偏差を表すが，探索領域の設定の段階では左右同じ（すなわち $s_i=s_i^L=s_i^R$）とする．また式（10）の a は図1の縦軸の値であり，システムパラメータであるが，その設定方法は3．5 節にて述べる．探索領域を図1 に例示する．

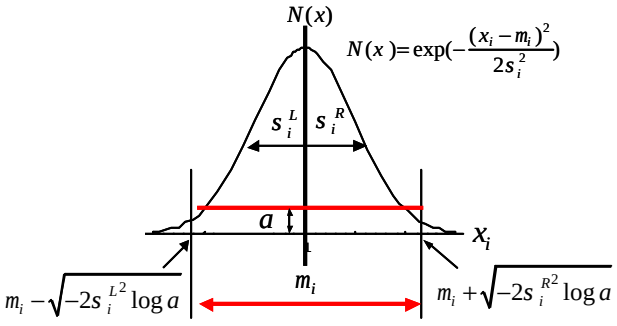

Fig.1 The active search domain of *i*-th design variable

3．3　最良値の保存　ARPSOでは探索領域が直前の探索における探索点の情報を利用して決定されるため，図2 に示すように探索領域が探索回数毎に変更される．図2中の m_i^k は k 回目の探索における i 番目の設計変数の平均（すなわち探索領域の中心）を表す．そのため，過去の探索において目的関数を最良にする探索点の設計変数の情報も失われてしまう．そこで，過去の探索において目的関数を最良にする探索点は必ず探索領域に入るように以下の操作を行う．なお以下の記述において x_i^{best} とは，p_g の i 番目の成分を表す．

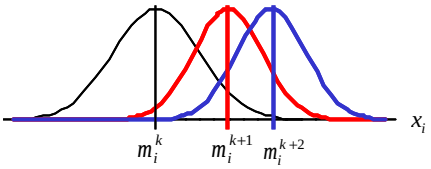

Fig.2 Variation of the active search domain

(1) $m_i+\sqrt{-2s_i^{R2}\log a} < x_i^{best}$ のとき

これは，図3に示すように x_i^{best} が探索領域外の右側に位置したときである．なお図3 中において実線は元の探索領域を示している．

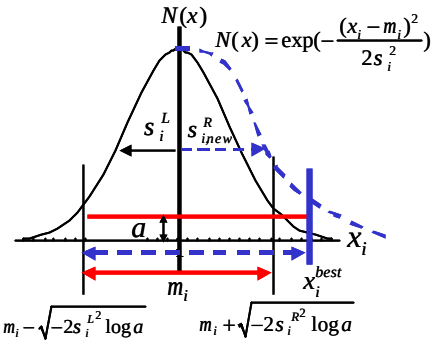

Fig.3 New active search domain by changing standard deviation

このとき，

$$a=\exp\left(-\frac{(x_i^{best}-m_i)^2}{2s_{inew}^{R\ 2}}\right) \qquad (11)$$

として，探索領域を決める新しい標準偏差 s_{inew}^R を求める．

$$s_{inew}^R=\sqrt{-\frac{(x_i^{best}-m_i)^2}{2\log a}} \qquad (12)$$

そして，新しい探索領域を以下のように設定する．

$$m_i-\sqrt{-2s_i^{L2}\log a} \le x_i \le m_i+\sqrt{-2s_{inew}^{R\ 2}\log a} \qquad (13)$$

図3 中の破線は最良値を保存することにより得られた新しい探索領域を示してる．

(2) $x_i^{best} < m_i-\sqrt{-2s_i^{L2}\log a}$ のとき

これは上記と逆の場合であり，x_i^{best} が探索領域外の左側に位置したときである．同様の手順で左側の新しい標準偏差 s_{inew}^L を求め，新しい探索領域を設定すればよい．つまり

$$s_{inew}^L=\sqrt{-\frac{(x_i^{best}-m_i)^2}{2\log a}} \qquad (14)$$

より，新しい探索領域は以下のようになる．

$$m_i - \sqrt{-2s_{inew}^{L\,2}\log a} \le x_i \le m_i + \sqrt{-2s_i^{R\,2}\log a} \quad (15)$$

3．4　側面制約条件への対処　図2に示したように，探索領域の中心は探索回数毎に変化することがあり，その結果，例えば図4 のように側面制約条件を破ってしまうことがある．（図4 の例では上限値を破ってしまった例を示している．）

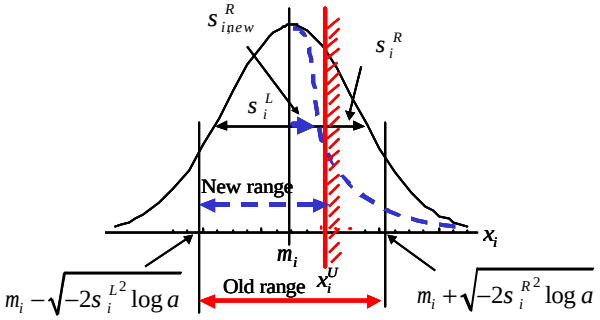

Fig.4 New active search domain considering the side constraint

図4 のような場合，新しい右側の標準偏差 s_{inew}^R を以下のように求める．

$$s_{inew}^R = \sqrt{-\frac{(x_i^U - m_i)^2}{2\log a}} \quad (16)$$

そして式(16)を式（13）に代入して新しい探索領域を設定する．一方，探索領域が下限値を下回った場合も同様な操作を行えばよく，左側の新しい標準偏差 s_{inew}^L を

$$s_{inew}^L = \sqrt{-\frac{(x_i^L - m_i)^2}{2\log a}} \quad (17)$$

として求め，式（17）を式（15）に代入して新しい探索領域を設定する．

3．5　システムパラメータの設定について　探索領域を決めるものは，各設計変数の平均と標準偏差と正規分布を基調とした図1の縦軸の値 a である．正規分布を基調としているため，a の最小値 $a_{\min}$ はゼロに近い正値を選べばよいが，問題となるのは a の最大値 $a_{\max}$ である．これは $a_{\max}$ を1 に近く選ぶと，探索終盤における探索領域が極端に小さくなりすぎ，有効な探索領域を確保できなくなるためである．また任意の値を設定することも可能であるが，アルゴリズムに複数のパラメータが存在すれば，パラメータの設定に設計者は多くの時間を割かれることになる．パラメータの推奨値やパラメータ決定の方針が立てられれば，設計者はパラメータの設定作業から解放される．そこで $a_{\max}$ の値について考える．

はじめに有効な探索範囲の確保という意味で i 番目の設計変数の標準偏差の最小値 $s_{i,\min}$ を決める．この $s_{i,\min}$ は側面制約条件を利用して

$$s_{i,\min} = e_1(x_i^U - x_i^L) \quad (18)$$

で決定するものとする．また，探索が十分進んだ段階では探索点は集中化するため，有効な探索領域は

$$e_2(x_i^U - x_i^L) \quad (19)$$

となるとする．式（18），（19）において e_1 は $s_{i,\min}$ を決めるパラメータであり，e_2 は探索が十分進んだ時の探索領域を決めるパラメータで，共に[0，1]の値である．式（10）と式（19）より

$$2\sqrt{-2s_{i,\min}^{\,2}\log a_{\max}} = e_2(x_i^U - x_i^L) \quad (20)$$

が成立する．ただし

$$s_{i,\min} = s_i^R = s_i^L \quad (21)$$

とおいた．式（20）に式（18）を代入することにより $a_{\max}$ は次式で表すことができる．

$$a_{\max} = \exp\left(-\frac{1}{8}\left(\frac{e_2}{e_1}\right)^2\right) \quad (22)$$

つまり $a_{\max}$ は e_1 と e_2 の比率のみから決定できることになる．そのため，設計者は具体的な e_1 と e_2 の値を考えるのではなく，$0 < e_2/e_1 \le 1$ となる比率のみを考えればよく，$a_{\max}$ は確定的な値として求まる．また数値実験の結果，$0.5 \le e_2/e_1 \le 1$ の程度がよいと思われる．

3．5　探索回数に応じた探索領域の設定　探索領域を決める一つの要素である a の最大値と最小値の設定については前節にて述べた．ここでは，a の値の変化について考える．一般に進化的計算と称される多点探索法では，探索が進むにつれ，全探索点が目的関数を最も良くする探索点に集中する．これに伴い有効な探索領域は，探索回数に応じて狭くなると考えるのが自然である．そこで，a の値を探索回数を考慮した以下の式を用いて変更する．

$$a = a_{\min} + \frac{(a_{\max} - a_{\min})}{k_{\max}} \times k \quad (23)$$

式（23）において k は探索回数，$k_{\max}$ は最大探索回数を表す．式（23）は基本的に式（2）の慣性項 w が線形的に減少するというものとは逆の考えであり，探索開始のときは比較的大きな探索領域が設定でき，探索回数の増加するにつれて有効な探索領域の絞込みを行うということを考慮した式である．

3．7　挙動制約条件の取扱い　挙動制約条件はペナルティ関数として扱い，事実上の無制約最適化問題として大域的最適解を求めることとした．

$$F(\boldsymbol{x}) = f(\boldsymbol{x}) + r \times penalty \rightarrow \min \quad (24)$$

式（24）において $penalty$ は，挙動制約条件の関数であり，r はペナルティ係数を表す．r の値は，目的関数と挙動制約条件の値の相対的な関係に左右され，制

約条件が正規化されていれば，目的関数と同じオーダーの値が適当であるとされている[6]．しかし挙動制約条件は一般に設計変数の陰な関数として表されることが多く[7]，適当な正規化も探索開始時には難しいため，事前の設定は困難なものと考える．また関数の感度を利用しない多点探索法では一般に外点ペナルティ関数法が用いられることが多く，その方法では $penalty$ の項は

$$penalty = \sum_{j=1}^{ncon} \max[0, g_j(\boldsymbol{x})] \tag{25}$$

もしくはその二乗となる．探索点が挙動制約条件をわずかに破った場合に式（25）を用いれば，式（24）の r を極めて大きな値に設定しなければペナルティの効果がないため，別途ペナルティ関数とペナルティ係数の決定法を考える必要がある．

そこで本論文では以下に示すペナルティ係数の決定法とペナルティ関数を用いるものとした．

$$r = \left(1 + \left|f(\boldsymbol{x})\right|\right)^q \tag{26}$$

$$\begin{aligned} &if \quad g_j(\boldsymbol{x}) > 0 \quad then \\ &\quad penalty = \sum_{j=1} \exp\left(1 + g_j(\boldsymbol{x})\right) \\ &else \\ &\quad penalty = 0 \\ &endif \end{aligned} \tag{27}$$

式（26）の q は1 以上の実数である．本論文では $q=2$ を用いた．式（26）を用いることで，挙動制約条件を違反した探索点の目的関数値を利用してペナルティ係数が自動的に決められ，また式（27）から，挙動制約条件の違反量を反映させることができる．

3.8　ARGAとの相違と提案方法の特徴　ARPSOでは，基本的にはARGAを参考としているため，探索領域の設定や最良値の保存，側面制約条件の対処による探索領域の変更は同じである．PSOが連続変数を直接的に扱うことができるのに対し，ARGAが基本的には連続変数を0 - 1 の二進数で表現しなければならないという点を考慮すると，その基本的な性質は大きく異なる．またARGAと以下の点で相違点を持つ．

（1）システムパラメータが極めて少ない[8]．また探索が十分進んだ段階で有効な探索領域を確保するためのシステムパラメータ $a_{\max}$ は e_1 と e_2 の比率のみから決定でき，確定的な値として求まる．

（2）式（23）を利用することにより，各設計変数の有効な探索領域は探索回数が考慮され，$a_{\min}$ の設定次第では探索開始時は比較的大きな探索領域を取ることも可能であり，探索回数の増加に伴い，探索領域が徐々に小さくなる．

（3）ペナルティ関数，ペナルティ係数の決定法が異なる．

4　アルゴリズム

ARPSOでは2章で述べたPSOの基本的なアルゴリズムの（STEP1）～（STEP5）までは同じため，それ以降についで記述する．

（STEP6）探索回数 k を $k=k+1$ とする．各設計変数の平均と標準偏差を求める．左右の標準偏差は同じとする．（$s_i = s_i^L = s_i^R$）

（STEP7）各設計変数の標準偏差をチェックする．もし式（18）による標準偏差の最小値を下回っている場合は，標準偏差の最小値とする．

$$s_i^L < s_{i,\min} \Rightarrow s_i^L = s_{i,\min} \tag{28}$$

$$s_i^R < s_{i,\min} \Rightarrow s_i^R = s_{i,\min} \tag{29}$$

（STEP8）式（23）による a の更新

（STEP9）式（10）による探索領域の設定．

（STEP10）今までの探索おける最良値 p_g が探索領域になければ式（12），（13）（もしくは式（14），（15））による探索領域の変更．

（STEP11）側面制約条件を満足していなければ式（16）（もしくは式（17））により標準偏差を再度計算しなおし，式（13）（もしくは式（15））による探索領域の変更．

（STEP12）式（1），（2）を用いて探索点を更新．

（STEP13）式（4）による慣性項 w の更新．

（STEP14）最大探索回数 $k_{\max}$ 以下ならSTEP6へ戻る．そうでなければ，探索終了する．

5　数値計算例

数値計算例を通じて，本論文で提案する方法の有効性を検討する．探索領域の変動を可視化するため，2変数問題を扱う．また以下の数値計算例では，有効探索領域を確保するための式（18）の e_1 は0.01（側面制約条件の1%），式（22）の e_2/e_1 は1と設定した．また式（23）の $a_{\min}$ は $a_{\min} = 1.0\times10^{-5}$ とした．また文献（9）によれば，PS0における探索点数は20～30が適切であるとされているため，問題に応じてこれらの範囲で個体数を変化させた．

5.1　無制約最適化問題（Schwefel関数）　次の関数の大域的最適解を求める．

$$f(\boldsymbol{x}) = 418.9829ndv + \sum_{i=1}^{ndv} \left\{ -x_i \sin\sqrt{|x_i|} \right\} \rightarrow \min \tag{30}$$

$$-500 \le \boldsymbol{x} \le 500 \tag{31}$$

大域的最適解は $x_{i,G} = 420.9687$ （$i = 1,2,\cdots,ndv$）であ

り，そのときの目的関数値は0 である．二次元の場合の関数の様子，等高線と大域的最適解を図5 に示す．

この問題は設計領域が広く，多くの局所的最適解が存在し，大域的最適解の探索には非常に多くの計算回数を要する．探索点数を２０，最大探索回数を１００とし，初期的に以下の範囲に探索点を与えて，大域的最適解の探索を行った．（図5 右の四角の領域）

$$-100 \le x \le 100 \qquad (32)$$

探索領域の変更の様子の一例を図6 に示す．なお図6 中において●は最良探索点を表す．また図7 にARPSO の最良値，g-best モデルの p_g^k と最良値保存モデルの p_g の目的関数値の履歴を示す．図8，9 には，各モデルの探索回数と各設計変数の標準偏差の和（ＡＲＰＳＯ：$s_i = s_i^R + s_i^L$，それ以外：$2s_i$）の関係を示す．各モデルの特徴を比較するため図7，8，9 について考える．

g-best モデルでは p_g^k が探索回数毎に更新されるため，探索初期段階では各設計変数の標準偏差が大きくなっていることが図8，9 よりわかる．また図7 より探索が十分進めば p_g^k は大域的最適解へ到達しているものの，図8，9 からわかるように各設計変数の標準偏差が大きい．これは集団が多様性を維持していることを意味する．しかし感度を利用しない進化的計算法と称される方法では，一般に探索が十分進めば，各設計変数の標準偏差が小さくなることにより，ある一つの最適解に集団が向うことが望ましい．つまりg-best モデルでは探索が十分進んだ段階でも設計領域全体に探索点がばらついていることを意味しており，探索が進んだ段階での探索点の集中化がおきていないことを示す結果である．

最良値保存モデルでは，g-best モデルと比べれば p_g が比較的早期に大域的最適解見つけており，また図8，9から標準偏差もg-best モデルと比較した場合，全体的に小さな値をとっていることがわかる．これは探索点の集中化が起きていることを意味している．しかし探索点が集中化しているものの，探索領域の絞込みはできない．

一方，ARPSO では最良値保存モデルよりも早期に大域的最適解を見つけており，各設計変数の標準偏差も小さいことがわかる．つまりARPSO は探索点の多様性が他のモデルに比べれば小さいにもかかわらず，設定された探索領域内で多様性を維持しつつ，集中的に探索を行っており，大域的最適解を見つけることができたということを意味している．これはARPSO が p_g （k 回目までの最良探索点）と p_g^k （k 回目における最良探索点）を併用しながら探索をしているためである．１０回試行した結果を表1 に示す．なお，g-best モデルの場合は，１ 度のみしか大域的最適解へ到達しなかったため，表1からは削除してある．

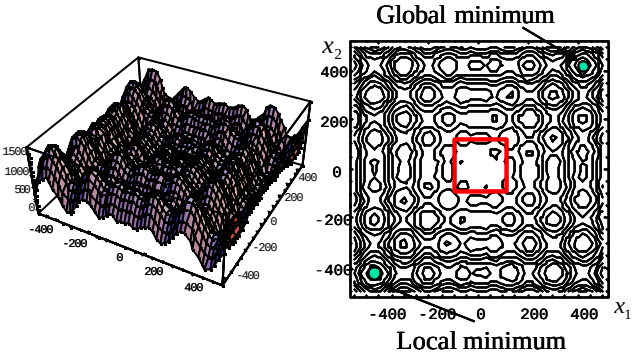

Fig.5 Behavior and contour of objective function

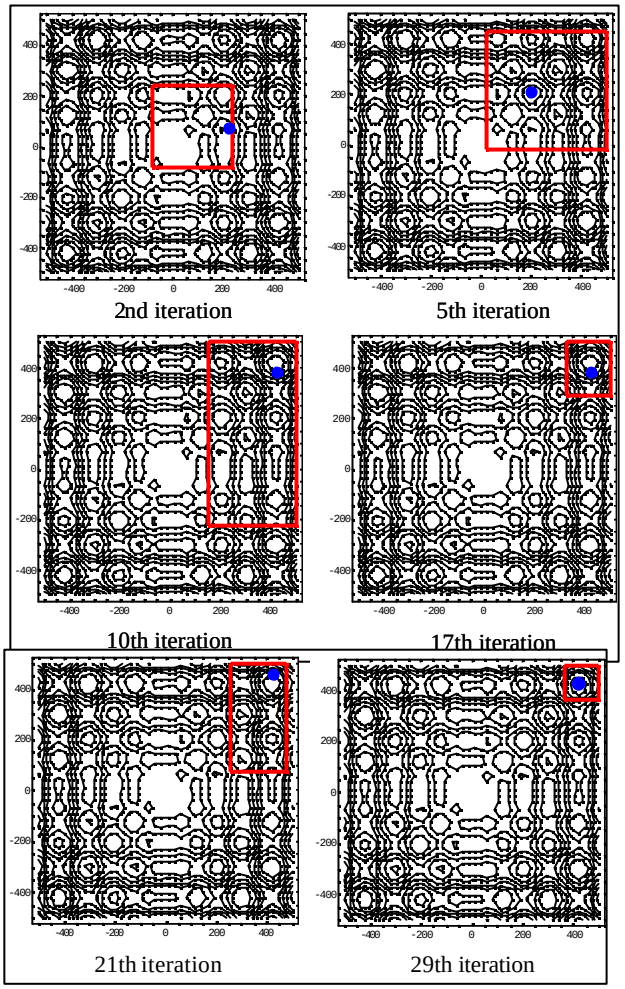

Fig.6 Change of search domain through search process

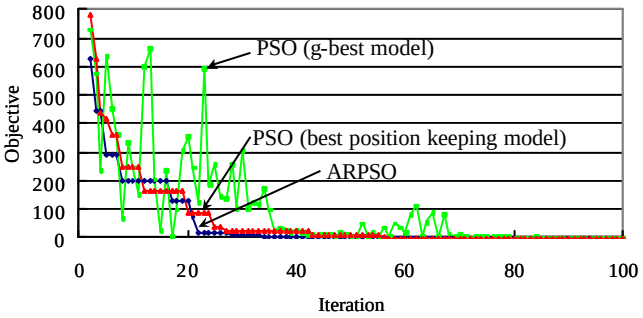

Fig.7 Convergence of objective function

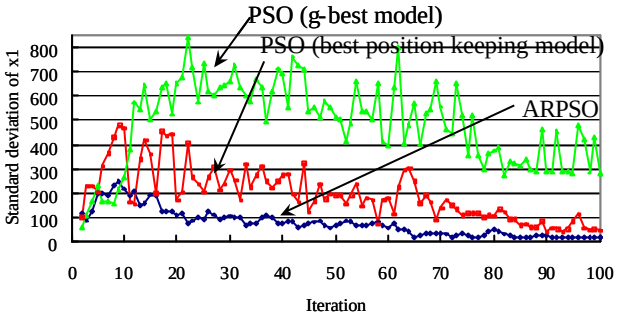

Fig.8 Standard deviation of design variable x_1

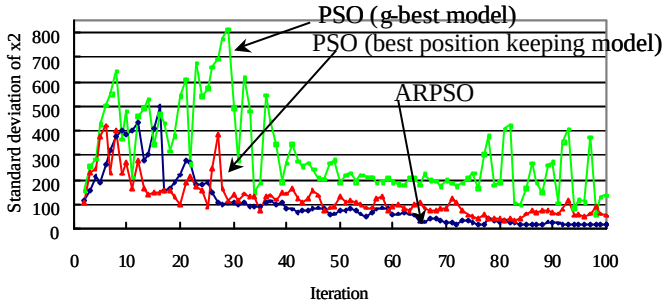

Fig.9 Standard deviation of design variable x_2

Table 1 Comparison of result

	ARPSO	best position
Best objective	2.545E-05	2.792E-05
Worst objective	2.758E-05	4.407E-03
Mean value of objective	2.593E-05	9.504E-04
Standard Deviation of objective	7.844E-07	1.282E-03
Average of function call	1488	1894

5.2 ベンチマーク問題における比較 5.1節の問題からわかるように，g-bestモデルは他のモデルよりも明らかに探索能力は低い．そこで最良値保存モデルとARPSOを，いくつかのベンチマーク問題に適用した．それらの結果を表2，3に示す．（用いたベンチマーク問題は付録に記載．）設計変数の数はすべて10とし，10回の試行を行った．また探索点数は30，最大探索回数を500とした．各表より以下のことがわかる．

(1) 計算回数（ファンクションコール）が明らかに少ない．よって効率的な探索が可能である．

(2) 目的関数値の平均および標準偏差が小さい．これは10回の試行で，安定的かつ精度の高い大域的最適解が得られたことを意味する．

5.3 コイルバネの重量最小化問題 文献(10)のコイルバネの重量最小化問題を考える．設計変数はワイヤの直径 $d(=x_1)$，コイルの平均直径 $D(=x_2)$，コイルの巻数 $N(=x_3)$ であり，すべて連続変数である．最適設計問題は次のように定式化される．

$$f(\boldsymbol{x}) = (2+x_3)x_1^2 x_2 \rightarrow \min \quad (33)$$

$$g_1(\boldsymbol{x}) = 1 - x_2^3 x_3 / (71785 x_1^4) \le 0 \quad (34)$$

$$g_2(\boldsymbol{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \le 0 \quad (35)$$

$$g_3(\boldsymbol{x}) = 1 - 140.45 x_1 / (x_2^2 x_3) \le 0 \quad (36)$$

$$g_4(\boldsymbol{x}) = (x_1 + x_2)/1.5 - 1 \le 0 \quad (37)$$

$$0.05 \le x_1 \le 2.00 \quad (38)$$

$$0.25 \le x_2 \le 1.30 \quad (39)$$

$$2.00 \le x_3 \le 15.0 \quad (40)$$

高々3変数の問題であるが，実行可能領域内の最適解を見つけにくい問題であり，さらに多峰性が激しいため，非常に多くの計算を要するとされている[11]．そのため多点探索型最適化手法のベンチマーク問題としてもよく用いられている[11-13]．探索点の数を20，最大探索回数を500とし，過去の研究と比較するために11回試行したときの結果を表4に示す．またすべての試行結果を表5に示す．これらの表より，現段階では最もよい目的関数値が得られており，さらに標準偏差等も小さく，安定的に最適解が得られていることがわかり，ARPSOの有効性の一端を確認できる．

6 結言

本論文では，筆者らの一人が開発した領域適応型遺伝的アルゴリズムを参考に，領域適応型Particle Swarm Optimization(ARPSO)を提案した．ARPSOでは探索領域が各設計変数の平均と標準偏差を用いて適宜可変する方法であり，探索が進むにつれ，探索領域を絞り込むことができるようなパラメータ a を導入した．このパ

Table 2 Result of 2n minima function

	ARPSO	best position
Best objective	-391.66166	-391.65783
Worst objective	-391.6616	-390.98815
Mean value of objective	-391.66165	-391.5102
Standard Deviation of objective	1.83E-05	2.00E-01
Average of function call	6879	13014

Table 3 Result of griewank function

	ARPSO	best position
Best objective	3.50E-09	8.93E-04
Worst objective	9.83E-08	7.30E-02
Mean value of objective	2.78E-08	2.91E-02
Standard Deviation of objective	3.16E-08	2.52E-02
Average of function call	7557	14799

Table 4 Comparison of results

Design Variables	Best solutions found				
	Arora[10]	Coello[11]	Ray[12]	Hu[13]	This study
$x1\ (d)$	0.053396	0.05148	0.050417	0.051466	0.051679
$x2\ (D)$	0.39918	0.351661	0.321532	0.351384	0.356477
$x3\ (N)$	9.1854	11.632201	13.979915	11.608659	11.299395
$g1(\boldsymbol{x})$	0.000019	-0.00208	-0.001926	-0.003336	-0.000037
$g2(\boldsymbol{x})$	-0.000018	-0.00011	-0.012944	-0.00011	-0.000008
$g3(\boldsymbol{x})$	-4.123832	-4.026318	-3.89943	-4.026318	-4.054976
$g4(\boldsymbol{x})$	-0.698283	-0.731239	-0.752034	-0.731324	-0.727895
$f(\boldsymbol{x})$	0.01273	0.012705	0.01306	0.012667	0.012661
Function Call	N/A	900000	1291	N/A	5804
Averege of $f(\boldsymbol{x})$	N/A	0.012769	0.013436	0.012719	0.012675
Worst of $f(\boldsymbol{x})$	N/A	0.012822	0.01358	N/A	0.012696
Standard Deviation of	N/A	3.9390E-05	N/A	6.4660E-05	1.1740E-05

Table 5 Results through 11 trials

Trial	x_1	x_2	x_3	$g_1(\boldsymbol{x})$	$g_2(\boldsymbol{x})$	$g_3(\boldsymbol{x})$	$g_4(\boldsymbol{x})$	obj.
1	0.0516792	0.3564771	11.2993956	-0.0000374	-0.0000086	-4.0549763	-0.7278958	0.0126618
2	0.0519954	0.3641189	10.8647657	-0.0000462	-0.0000295	-4.0696722	-0.7225905	0.0126641
3	0.0520007	0.3642496	10.8581928	-0.0001074	-0.0000276	-4.0696214	-0.7224998	0.0126648
4	0.0511883	0.3447864	12.0202057	-0.0000233	-0.0000064	-4.0312974	-0.7360169	0.0126662
5	0.0522840	0.3711950	10.4843893	-0.0000062	-0.0000126	-4.0832736	-0.7176806	0.0126680
6	0.0509491	0.3391693	12.3939067	-0.0000926	-0.0000170	-4.0190010	-0.7399210	0.0126727
7	0.0508524	0.3369097	12.5485456	-0.0000393	-0.0000287	-4.0143310	-0.7414919	0.0126752
8	0.0527666	0.3831984	9.8872390	-0.0000902	-0.0000002	-4.1045623	-0.7093567	0.0126830
9	0.0528525	0.3853432	9.7858349	-0.0000198	-0.0000307	-4.1085073	-0.7078696	0.0126864
10	0.0529021	0.3865957	9.7274183	-0.0000172	-0.0000247	-4.1107290	-0.7070015	0.0126883
11	0.0503256	0.3247856	13.4354849	-0.0000406	-0.0000184	-3.9872798	-0.7499258	0.0126968

ラメータの上限値 $a_{\max}$ は確定的に決めることができる．また挙動制約条件を含む，より一般的な最適化問題に対しては，挙動制約条件をペナルティ関数として扱い，事実上の無制約最適化問題に変換する方法を用いた．本論文で取り扱った数値計算例では，ARPSOは計算効率がよく，さらに精度の高い大域的最適解の探索が可能であることが判明した．

本研究を行うにあたり，適切なご助言を頂いた山川宏先生（早稲田大学），中山弘隆先生（甲南大学），杉本博之先生（北海学園大学）に感謝したい．

参考文献

(1)Parsopoulos, K.E.,Vrahatis M.N., *Recent approaches to global optimization problems through Particle Swarm Optimization*, Natural Computing,**1**,(2002),pp.235-306.

(2)Kitayama,S., et al., Basic Examination on Particle Swarm Optimization and Its Application to the Mixed Design Variable Problems, ***N**ihon **K**ikai **G**akkai **R**onbunshu A(**Transactions of the Japan Society of Mechanical Engineers**, **Series A)***, **71**-706, (2005), 968-975. (in Japanese).

(3) Parsopoulos, K.E.,Vrahatis M.N., *On the computation of all global minmizers through particle swarm optimization*, IEEE trans. on Evolutionary Computation, Vol.8, No.3,(2004),pp.211-224.

(4)Parsopoulos, K.E.,Vrahatis M.N., *UPSO: A Unified Particle Swarm Optimization Scheme*, Lecture Series on Computer and Computational Science, Vol.1 (2004), pp.868-873.

(5)Arakawa,M., Hagiwara,I., Development of Adaptive Real Range (ARRange) Genetic Algorithms,***N**ihon **K**ikai **G**akkai **R**onbunshu C(**Transactions of the Japan Society of Mechanical Engineers**, **Series C)***, **63**-616,(1997),4216-4223. (in Japanese).

(6)Sugimoto, H., Application of GA to the Industrial Design, Mathematical Sciences, No.353, (1992), 45-60.(in Japanese)

(7)Yamakawa, H., Optimum Design, (1993), Baifukan.

(8)Arakawa, M., Hagiwara, I., Development of Adaptive Range Genetic Algorithms (Proposal of the new operators for efficient and high accurate solutions), ***N**ihon **K**ikai **G**akkai **R**onbunshu C(**Transactions of the Japan Society of Mechanical Engineers**, **Series C)***, **65**-638,(1999),4156-4163. (in Japanese).

(9)Schutte, F., Groenwold, A., A., A Study of Global Optimization Using Particle Swarm, Journal of Global Optimization, Vol.31, (2005), pp.93-108.

(10) Arora, J.S., *Introduction to Optimum Design*, (1989), McGraw-Hill, New York.

(11) Coello Coello, C.A., Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems, *Computers in Industry*, Vol.41, (2000), pp.113-127.

(12) Ray, T., Saini, P., Engineering Design Optimization Using Swarm with an Intelligent Information Sharing among Individuals, *Engineering Optimization*, Vol.33, (2001), pp.735-748.

(13) Hu, X. H., et. al., Engineering Optimization with Particle Swarm, *IEEE Swarm Intelligence Symposium*, (2003), pp.53-57.

付録

1.2n-minima関数

$$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{ndv}\left(x_i^4 - 16x_i^2 + 5x_i\right) \to \min$$

$$-5 \le \boldsymbol{x} \le 5$$

大域的最適解は $\boldsymbol{x}_G = (-2.90354,\cdots,-2.90354)^T$ **であり，設計変数の数が10のとき，**

$$f(\boldsymbol{x}_G) = -391.6616$$

2.Griewank関数

$$f(\boldsymbol{x}) = 1 + \frac{1}{D}\sum_{i=1}^{ndv} x_i^2 - \prod_{i=1}^{ndv}\cos\left(\frac{x_i}{\sqrt{i}}\right) \to \min$$

$$-10 \le \boldsymbol{x} \le 10$$

大域的最適解と目的関数： $\boldsymbol{x}_G = (0,0,\cdots,0)^T$ ， $f(\boldsymbol{x}_G) = 0$

本論文では $D = 400$ **とした．**