数理解析研究所講究録 1477 巻 2006 年 1-10

確率的比較を用いた制約付き最適化法「確率的変換法」の提案

広島修道大学商学部　阪井　節子 (Setsuko Sakai)
Faculty of Commercial Science, Hiroshima Shudo University
広島市立大学情報科学部　高濱　徹行 (Tetsuyuki Takahama)
Faculty of Information Sciences, Hiroshima City University

1 まえがき

与えられた制約の下で目的関数を最適化する制約付き最適化問題、特に制約付き非線形最適化問題は実世界に頻繁に出現する重要な最適化問題である．制約付き非線形最適法問題の解法としては，逐次2次計画法，射影法，一般縮小勾配法などの効率的な方法が存在する．しかし，これらの方法は目的関数の微分可能性や制約領域の凸性など問題に対して幾つかの条件を仮定している．したがって，これらの方法を様々な分野における問題に広く適用することは困難である．

これに対して，目的関数の値だけを利用して制約のない非線形最適化問題を解決する方法が提案されている．この方法は直接探索法 (direct search method) と呼ばれ，様々な問題に適用することができるが，制約付き非線形最適法問題を直接解くことはできない．本研究では，直接探索法を制約付き非線形最適法問題に適用する方法について考察する．

制約付き最適化問題を直接探索法によって解く際には，目的関数だけではなく，一般に複数の制約も最適化する必要がある．制約付き最適化のためには、目的関数の最適化と制約の最適化という2種類の最適化が含まれるため，2つの最適化のバランスを適切に取ることが必要となる．制約を扱う方法は，同時に最適化する目的の数に基づき，以下のように分類できる．

(1) 目的関数のみを最適化する方法

一つ以上の実行可能解を初期探索点として準備し，制約を満足する探索点のみを考慮してゆくことにより，制約の最適化を省略する方法であり，death penalty 法とも呼ばれる．探索の過程で得られた点が制約を満足しない場合には，単純に無視されるか，制約を満足するように修正される．例えば，遺伝的アルゴリズム (Genetic Algorithm, GA) において，制約を満足した探索点を参照して制約を満足しない探索点を修正する方法が提案されている [1, 2, 3]．また，パーティクルスォーム最適化 (Particle Swarm Optimization, PSO)[4, 5, 6] において，制約を満足しない探索点を無視し，既知の制約を満足する探索点に置換する方法も提案されている [7]．これらの方法は制約領域が比較的広い場合には有効である．しかし，実際には制約の厳しい問題も多く，特に等式制約を含む問題では，初期点を準備したり探索点を修正することは不可能に近い．

(2) 目的関数と制約逸脱度 (constraint violation) の荷重和を最適化する方法

複数の制約条件を組み合わせて制約逸脱度を定義し，目的関数と制約逸脱度の荷重和を求め，その荷重和の一目的最適化問題として解く方法である．制約逸脱度は目的関数に対するペナルティと考えられるため，この方法は一般にペナルティ関数法 (penalty function method) と呼ばれ，目的関数と制約逸脱度の荷重和は拡張目的関数 (extended objective function) と呼ばれている．ペナルティ関数法では，制約逸脱度の強さを調整するための荷重であるペナルティ係数 (penalty coefficient) を適切に選択することが困難であるという問題点がある．ペナルティ係数が大きいと，制約を満足する解は得られるが，目的関数の最適化が不十分になり，質の高い解を得ることが困難になる．逆にペナルティ係数が小さいと，目的関数は最適化されるが，制約の最適化が不十分になり，実行可能解を得ることが困難になる．ペナルティ係数を動的に調整する方法もあるが，適切な調整方法は問題に依存するため，一般的な調整方法を実現するのは困難であるという問題がある．

(3) 目的関数と制約逸脱度を辞書式比較により最適化する方法

目的関数と制約逸脱度を分離して扱い，制約逸脱度を優先する辞書式比較により一目的化して解く方法である．例えば，GA において辞書式比較を実現する拡張目的関数を利用する方法が提案されている [8]．これは，制約を満足しない探索点の拡張目的関数値を，集団中の制約を満足する探索点における最悪の目的関数値と制約逸脱度との和として与える方法である．進化的戦略 (Evolutionary Strategy, ES) において，単に制約逸脱度を優先するのではなく，ある確率で制約逸脱度を無視し目的関数のみで比較を行うという拡張された辞書式比較により最適化を行う方法が提案されている [9]．これにより，制約条件が確率的に緩和され，質の高い実行可能解が得られることが示されている．より一般的な方法として，直接探索法全般に対して，制約条件を緩和することができる辞書式比較である α レベル比較を使用する α 制約法 [10, 11, 12, 13, 14, 15, 16, 17, 18] および ϵ 比較を使用する ϵ 制約法 [19, 20] が提案されている．α 制約法および ϵ 制約法は，直接探索法における比較演算子を α レベル比較および ϵ レベル比較に置換することにより，制約のない問題に対する最適化アルゴリズムを制約付きの最適化アルゴリズムに変換する方法，すなわちアルゴリズム変換法である．α および ϵ 制約法は，制約条件を緩和することにより，等式制約を含むような制約条件の厳しい問題に対しても適用することができる．

(4) 目的関数と各制約の多目的問題として解く方法

目的関数と一般に複数の制約関数を多目的最適化問題として解く方法である [21, 22]．制約が複雑な問題に有効であると期待されるが，多目的最適化問題は一目的問題と比較すると非常に困難な問題であり，一般に多くの計算量を必要とするという問題点がある．

本研究では，確率的な辞書比較という考えを一般化し，確率に基づく比較を定義し，この定義に基づく新しいアルゴリズム変換法を提案する．

2 制約付き最適化問題

2.1 定義

本論文では，次のような不等式制約，等式制約，上下限制約を持つ最適化問題 (P) を考える．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．

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

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，$g_j(\boldsymbol{x}) \leq 0$ 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, h 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．さらに，以下では全ての制約を満足する領域を F，上下限制約を満足する領域を S と呼ぶことにする．

2.2 制約逸脱度

初めに述べたように，制約付き最適化では，目的関数の最適化と制約の最適化という 2 面性が存在する．目的関数は $f(\boldsymbol{x})$ で明示的に与えられているため，ここでは，制約の最適化の指標となる制約逸脱度を定義する．制約逸脱度は，ペナルティ関数法で使用されているペナルティと同等のものを使用する．このとき，ある点 $\boldsymbol{x}$ における制約逸脱度 $\phi(\boldsymbol{x})$ は以下のように与えられる．

$$\phi(\boldsymbol{x}) = \max\{\max_j\{0, g_j(\boldsymbol{x})\}, \max_j |h_j(\boldsymbol{x})|\} \tag{2}$$

$$\phi(\boldsymbol{x}) = \sum_j ||max\{0, g_j(\boldsymbol{x})\}||^r + \sum_j ||h_j(\boldsymbol{x})||^r,\ (\text{ここで} r > 0 \text{である}) \tag{3}$$

これにより，制約付き最適化問題は以下のように定義できる．

$$(P^{\phi})\ \text{minimize}\quad f(\boldsymbol{x}) \qquad \text{subject to}\ \ \phi(\boldsymbol{x})=0 \tag{4}$$

3 確率的制約法

3.1 アルゴリズム変換法

本研究では，新しいアルゴリズム変換法として，確率的制約法 (probabilistic constrained method) を提案する．アルゴリズム変換法とは，ペナルティ関数法のように目的関数を変換して制約付き最適化問題を解くのではなく，制約のない問題に対するアルゴリズムを制約付き最適化アルゴリズムに変換する新しいタイプの変換法である．アルゴリズム変換法として，α および ϵ 変換法が提案されている．α および ϵ 変換法では，制約領域を拡大し，拡大された領域で最適値を探索することにより，目的関数の最適化と制約の最適化のバランスを取っている．これに対して，確率的制約法では，基本的には制約の最適化を優先した探索を行うが，ある確率で目的関数の最適化を優先した探索を行うことにより，両者のバランスを取ろうとする方法である．探索のバランスを確率的に取るために，ある確率で目的関数の大小関係を優先する確率的比較を定義する．

3.2 確率的比較

目的関数値と制約逸脱度の組 (f,ϕ) の集合上において，確率 p で目的関数の最適化を優先し，確率 $1-p$ で制約逸脱度を優先する確率比較 (probabilistic comparison) を定義する．

点 $\boldsymbol{x}_1,\boldsymbol{x}_2$ における関数値を f_1,f_2，制約逸脱度を ϕ_1,ϕ_2 とすると，通常の大小関係である $<,\leq$ に対応する関数値と制約逸脱度の組 (f_i,ϕ_i) 間の大小関係である確率比較 ${}_p<,{}_p\leq(0\leq p\leq 1)$ は以下のようになる．

$$(f_1,\phi_1)\,{}_p{<}\,(f_2,\phi_2) \Leftrightarrow \begin{cases} \phi_1<\phi_2, & f_1=f_2 \\ f_1<f_2, & \phi_1=\phi_2 \\ \phi_1<\phi_2 \ \text{w.p.}\ 1-p & \\ f_1<f_2 \ \text{w.p.}\ p & \end{cases}, \ \text{otherwise} \tag{5}$$

$$(f_1,\phi_1)\,{}_p{\leq}\,(f_2,\phi_2) \Leftrightarrow \begin{cases} \phi_1\leq\phi_2, & f_1=f_2 \\ f_1\leq f_2, & \phi_1=\phi_2 \\ \phi_1\leq\phi_2 \ \text{w.p.}\ 1-p & \\ f_1\leq f_2 \ \text{w.p.}\ p & \end{cases}, \ \text{otherwise} \tag{6}$$

ここで，p は一般に ϕ_1,ϕ_2,f_1,f_2 の関数，すなわち逸脱確率関数 (violation probability function) であり，制約の逸脱を許容する確率を決定する．

確率的比較において，${}_0<,{}_0\leq$ は制約を優先する辞書式比較と一致し，${}_1<,{}_1\leq$ は，目的関数を優先する辞書式比較と一致する．確率パラメータ p を小さくすれば，制約を満足する解が優先的に探索され，p を大きくすれば，目的関数の優れた解が優先的に探索される．したがって，p により制約の最適化と目的関数の最適化のバランスを調整することができる．

3.3 確率パラメータの制御

確率制約法は，直接探索法において，比較演算子を確率的比較に置換することにより，制約のない問題に対する最適化アルゴリズムを制約付きの問題に対する最適化アルゴリズムに変換するアルゴリズム変換法である．ここで，確率的比較により順序づけした最適化問題 (P_p) について考察する．

$$(P_p)\ \text{minimize}_{\,p}{<}\quad f(\boldsymbol{x}) \tag{7}$$

問題 (P_0) は，制約を優先して目的関数を最適化するため，得られる解が (P^ϕ) の解と一致することは自明である．しかし，実際に制約を優先する辞書式比較 $_0<$ で最適化を行うと，目的関数の最適化への探索が弱くなり，実行可能解は得られるが，目的関数については準最適解しか得られないことがある．このため本研究では，質の高い実行可能解を求めるために，初期に p を大きく，最終的には p を 0 にするような調整を行うことにより，最適化のバランスを取ることを提案する．すなわち，以下のような変換を行っていることになる．

$$(P) = (P^\phi) = (P_0) = \lim_{p\to 0}(P_p) \tag{8}$$

4 確率的制約パーティクルスォーム最適化 pPSO

4.1 パーティクルスォーム最適化

PSO による最適化を簡単に説明する．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．

$$pbest_i = \min_{\tau=0,1,\cdots,t} f(\boldsymbol{x}_i^\tau), \quad \boldsymbol{x}_i^* = \arg\min_{\tau=0,1,\cdots,t} f(\boldsymbol{x}_i^\tau) \tag{9}$$

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．

$$gbest = \min_i pbest_i, \quad \boldsymbol{x}_G^* = \arg\min_i f(\boldsymbol{x}_i^*) \tag{10}$$

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．

$$v_{ij}^{t+1} = wv_{ij}^t + c_1\, rand\,(x_{ij}^* - x_{ij}^t) + c_2\, rand\,(x_{Gj}^* - x_{ij}^t) \tag{11}$$

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 (11) 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．

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{12}$$

4.2 pPSO のアルゴリズム

確率的制約パーティクルスォーム最適化 (probabilistic constrained 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：確率的比較により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 (11),(12) 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．

pPSO のアルゴリズムを以下に C 言語風に記述する.

```
pPSO()
{
  Initialize Agents(0);
  Evaluate all x in Agents(0);
  x*_G=arg min_p< 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+c1rand(x*_ij−x_ij)+c2rand(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)_p< f(x*_i)) {
        if(f(x_i)_p< f(x*_G)) G=i;
        x*_i=x_i;
      } } } }
```

5 数値実験

ここでは，ペナルティ関数法を含んだ数多くの方法で解かれている制約付き最適化を取り上げる.

5.1 実験条件

本研究では，PSO に関するパラメータは，エージェント数 20，慣性重みのパラメータ $w^0 = 0.9, w^T = 0.4$，探索の重みパラメータ $c_1 = 2.0, c_2 = 2.0$，速度の最大値 V_{max_j} を各次元の探索領域の幅を基準に $0.2(u_j - l_j)$ とした．V_{max_j} が大きい場合には，制約領域外を探索する頻度が増加し，制約領域内の探索が十分に行われなくなるため，小さめの値を選択した.

確率的制約法における逸脱確率関数は，目的関数値に依存せず，制約逸脱度のみに依存する関数として，以下のように定義した.

$$p(\phi_1, \phi_2) = p_{\max} \exp(\beta(\phi_1 - \phi_2)/\phi_{\mathrm{w}}) \tag{13}$$

$$\phi_{\mathrm{w}} = \max_{k \in Agents(t)} \{\phi_k\} - \min_{k \in Agents(t)} \{\phi_k\} \tag{14}$$

ここで，p_{max} は逸脱の最大確率を指定するパラメータ，β は制約逸脱度の差異による確率の減衰を指定するパラメータ，ϕ_{w} はエージェント内における制約逸脱度の最大の差異であり，正規化のために用いている.

この定義では，探索が進むにしたがって，ϕ_{w} が 0 に近づくため，関数値が良くても制約逸脱度が大きい場合には「良い」と判定される確率が低くなる．このため，明示的に p あるいは p_{max} の値を 0 にする制御は行わず，以下の実験では，$p_{max} = 0.05, \beta = \log(0.1)$ と固定し，30 回の試行を行った.

5.2 Himmerblau の問題

Himmerblau の問題は以下のように定式化される.

$$\begin{aligned}
&\text{Minimize} \\
&f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \\
&\text{Subject to} \\
&g_1(\boldsymbol{x}) = 85.334407 + 0.0056858x_2x_5 + 0.00026x_1x_4 - 0.0022053x_3x_5, \\
&g_2(\boldsymbol{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2, \\
&g_3(\boldsymbol{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4, \\
&\quad 0 \le g_1(\boldsymbol{x}) \le 92, 90 \le g_2(\boldsymbol{x}) \le 110, 20 \le g_3(\boldsymbol{x}) \le 25, \\
&\quad 78 \le x_1 \le 102, 33 \le x_2 \le 45, 27 \le x_3, x_4, x_5 \le 45.
\end{aligned} \tag{15}$$

表 1 に実験結果を示す．pPSO 以外の結果は，文献 [23] に基づくものであり，MGA は多目的 GA と解の優越関係によるアルゴリズム，Gen は遺伝的アルゴリズムを利用したアルゴリズム，GRG は Generalized Reduced Gradient 法，Death は death penalty 法，その他はペナルティに基づくアルゴリズムであり，ペナルティ係数を固定する static penalty, 探索ステップ数によりペナルティ係数を変化させる dynamic penalty, simulated annealing のように温度によりペナルティ係数を変化させる annealing penalty, 複数の探索点の状態によりペナルティ係数を決定する adaptive penalty，解の集団と 2 種類のペナルティル係数のための集団を用いて共進化させる Coevolutionary penalty 法である．

各アルゴリズムによる試行中の最良値，平均値，最悪値および標準偏差を示した．なお，pPSO については，関数評価回数が 5,000 回の結果の下に 50,000 回の結果も示した．良い結果を示したアルゴリズムは pPSO, MGA, Co-evolutionary である．関数評価回数が 5,000 回の pPSO と MGA を比較すると，全ての項目で pPSO の方が優れている．また，評価回数が 50,000 回の pPSO と評価回数が 900,000 回の Co-evolutionary を比較しても全ての項目で pPSO の方が優れている．したがって，pPSO は効率よく探索を行える安定したアルゴリズムであるといえる．

表 1: Result of Himmerblau's problem

Algorithm	Best	Average	Worst	S.D.
pPSO (5,000)	**-31014.5953**	**-30996.5476**	**-30945.2652**	**18.7223**
(50,000)	**-31025.5591**	**-31025.4779**	**-31024.5841**	**0.1782**
MGA	-31005.7966	-30862.8735	-30721.0418	73.240
Gen	-30183.576	N/A	N/A	N/A
GRG	-30373.949	N/A	N/A	N/A
Co-evolutionary	-31020.859	-30984.2407	-30792.4077	73.6335
Static	-30790.2716	-30446.4618	-29834.3847	226.3428
Dynamic	-30903.877	-30539.9156	-30106.2498	200.035
Annealing	-30829.201	-30442.126	-29773.085	244.619
Adaptive	-30903.877	-30448.007	-29926.1544	249.485
Death	-30790.271	-30429.371	-29834.385	234.555

5.3 角材の溶接の設計

角材の剪断応力 (τ)，曲げ応力 ($sigma$)，台の座屈荷重 (P_c)，角材の端のたわみ (δ) などの制約の元でコストが最小となる角材の溶接を設計する．図 1 のように，4 つの決定変数 $h(x_1)$, $l(x_2)$, $t(x_3)$, $b(x_4)$ により設計する．

この問題は以下のように定式化される．

Minimize (16)

$$f(\boldsymbol{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x_2)$$

Subject to

$$g_1(\boldsymbol{x}) = \tau(\boldsymbol{x}) - \tau_{max} \le 0,\ g_2(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - \sigma_{max} \le 0,\ g_3(\boldsymbol{x}) = x_1 - x_4 \le 0,$$

$$g_4(\boldsymbol{x}) = 0.10471 x_1^2 + 0.04811 x_3 x_4 (14 + x_2) - 5 \le 0,\ g_5(\boldsymbol{x}) = 0.125 - x_1 \le 0,$$

$$g_6(\boldsymbol{x}) = \delta(\boldsymbol{x}) - \delta_{max} \le 0,\ g_7(\boldsymbol{x}) = P - P_c(\boldsymbol{x}) \le 0,\ 0.1 \le x_1, x_4 \le 2, 0.1 \le x_2, x_3 \le 10,$$

where

$$\tau = \sqrt{\tau'^2 + 2\tau'\tau'' \frac{x_2}{2R} + \tau''^2},\ \tau' = \frac{P}{\sqrt{2} x_1 x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right),$$

$$R = \sqrt{\frac{x_2^2 + (x_1 + x_3)^2}{4}},\ J = 2\sqrt{2} x_1 x_2 \left(\frac{x_2^2}{12} + \frac{(x_1 + x_3)^2}{4}\right),$$

$$\sigma(\boldsymbol{x})=\frac{6PL}{x_4x_3^2},\ \delta(\boldsymbol{x})=\frac{4PL^3}{Ex_3^3x_4},\ Pc(\boldsymbol{x})=\frac{4.013E\sqrt{x_3^2x_4^6/36}}{L^2}\left(1-\frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$

$$P=6000lb, L=14in, \delta_{max}=0.25in,\ E=30\times10^6psi, G=12\times10^6psi,$$

$$\tau_{max}=13600psi, \sigma_{max}=30000psi.$$

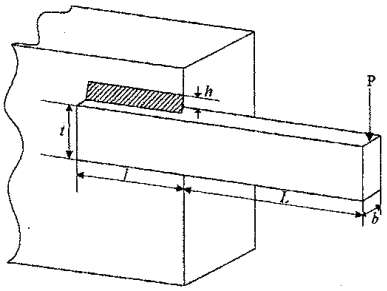

図 1: Welded beam design

表 2: Result of welded beam problem

Algorithm	Best	Average	Worst	S.D.
pPSO (5,000)	**1.7252**	**1.7393**	**1.8140**	**0.01891**
(50,000)	**1.7249**	**1.7249**	**1.7253**	**0.00011**
MGA	1.8245	1.9190	1.9950	0.05377
Co-evolutionary	1.7483	1.7720	1.7858	0.01122
Static	2.0469	2.9728	4.5741	0.6196
Dynamic	2.1062	3.1556	5.0359	0.7006
Annealing	2.0713	2.9533	4.1261	0.4902
Adaptive	1.9589	2.9898	4.84036	0.6515
Death	2.0821	3.1158	4.5138	0.6625

表 2 に実験結果を示す．良い結果を示したアルゴリズムは pPSO, MGA, Co-evolutionary である．関数評価回数が 5,000 回の pPSO と MGA を比較すると，全ての項目で pPSO の方が優れており，非常に精度の高い最良解を見つけている．また，pPSO と評価回数が 900,000 回の Co-evolutionary を比較すると，評価回数が 5,000 回の場合でも平均的には pPSO が優れており，評価回数が 50,000 回の場合には全ての項目で pPSO の方が優れている．したがって，pPSO は効率よく探索を行えるアルゴリズムであるといえる．

5.4 圧力容器の設計

半球状のキャップが両端に付いている円筒状の容器において，材料，形成，溶接に必要なコストを最小化する問題である．図 2 に示すように，T_s(シェルの厚み)，T_h(キャップの厚み)，R(内径)，L(円筒状の長さ)の 4 変数を設計する．このうち，T_s と T_h は利用可能な筒状鋼板の厚みから，0.0625 インチの整数倍である．

この問題は以下のように定式化される．表 3 に実験結果を示す．Deb は Genetic Adaptive Search，Kannan

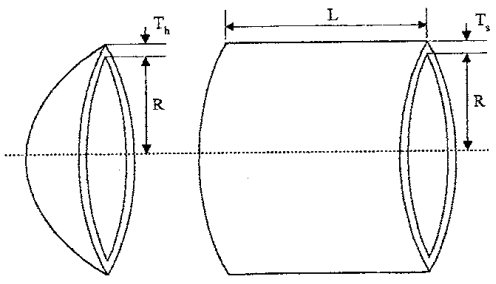

図 2: Pressure Vessel design

Minimize (17)

$$f(\boldsymbol{x})=0.6224x_1x_3x_4+1.7781x_2x_3^2$$
$$+3.1661x_1^2x_4+19.84x_1^2x_3$$

Subject to

$$g_1(\boldsymbol{x})=-x_1+0.0193x_3\le0,$$
$$g_2(\boldsymbol{x})=-x_2+0.00954x_3\le0,$$
$$g_3(\boldsymbol{x})=-\pi x_3^2x_4-4\pi/3x_3^3+1296000\le0,$$
$$g_4(\boldsymbol{x})=x_4-240\le0,$$
$$x_1,x_2=0.0625i, i\in\{1,2,\cdots,99\}, 10\le x_3,x_4\le200.$$

は拡張 Lagrangian Multiplier 法，Sandgen は Branch and Bound 法によるものである．

良い結果を示したアルゴリズムは pPSO, MGA, Co-evolutionary である．関数評価回数が 50,000 回の pPSO と MGA を比較すると，平均的には pPSO の方がかなり良質な解を見つけ，非常に精度の高い最良

解も見つけている．しかし，最悪解および標準偏差についてはやや MGA の方が良い結果となっているが，その差はわずかである．また，評価回数が 50,000 回の pPSO と評価回数が 900,000 回の Co-evolutionary を比較すると，最良値と平均値については pPSO の方が優れており，100,000 回の pPSO は平均値が 200 以上も優れている．したがって，pPSO は効率よく探索を行えるアルゴリズムであるといえる．

表 3: Result of Pressure Vessel problem

Algorithm	Best	Average	Worst	S.D.
pPSO (50,000)	**6059.7143**	**6154.9023**	6431.4942	114.375
(100,000)	**6059.7143**	**6092.5940**	**6204.3033**	29.9741
MGA	6069.3267	6263.7925	**6403.4500**	**97.9445**
Deb	6410.3811	N/A	N/A	N/A
Kannan	7198.0428	N/A	N/A	N/A
Sandgen	8129.1036	N/A	N/A	N/A
Co-evolutinary	6288.7445	6293.8432	6308.1497	**7.4133**
Static	6110.8117	6656.2616	7242.2035	320.8196
Dynamic	6213.6923	6691.5606	7445.6923	322.7647
Annealing	6127.4143	6660.8631	7380.4810	330.7516
Adaptive	6110.8117	6689.6049	7411.2532	330.4483
Death	6127.4143	6616.9333	7572.6591	358.8497

5.5 議論

上記の結果から，pPSO は少ない関数評価回数で非常に高精度の解を得られる方法であるといえる．特に，Welded Beam と Pressure Vessel では それぞれ 5,000 回と 50,000 回という少ない評価回数にも関わらず 900,000 回の評価を行なった Coevolutionary penalty 法の最良解より優れた解を発見しており，その探索能力の高さは評価に値すると考える．

また，pPSO は PSO に基づいており，非常に高速に動作するという点も重要である．例えば MGA では各世代毎に解の優越関係を調べるために集団中の任意の 2 対の比較を行う必要があり，そのコストは $N(N-1)/2$ であるが，PSO では各エージェントの最良値およびグループの最良値との比較のみであり，コストは最大 $2N$ である．また，親の選択が不要であり，世代交代のための操作が不要であることなどから遺伝的アルゴリズムと比較すると非常に高速に動作する．本研究で提案した pPSO は非常に高速に動作し，1GHz Pentium M のノートパソコンでは，Himmerblau 問題は 0.0070 秒，Welded Beam 問題は 0.0082 秒，Pressure Vessel 問題は 0.0529 秒という短時間で計算可能である．

6 確率的比較の効果

3 種類の制約付き最適化問題について，様々の方法と比較することにより，pPSO が効率の良い安定した制約付き最適化アルゴリズムであることを示した．ここでは，確率的比較のパラメータ $p_{\max} = 0.0, 0.02, 0.04, 0.05, 0.06, 0.08, 0.1$ と設定し，逸脱の最大確率を変化させることにより，探索結果にどのような影響があるかを調べる．より精密な結果を得るために，100 回の試行を行った．なお，$p_{\max} = 0$ は制約逸脱度を優先する比較により探索することを意味する．

表 4 に Himmerblau 問題の実験結果を示す．なお，関数評価回数は，5,000 回である．最良値，平均値，最悪値，標準偏差について全て $p_{\max} = 0.04$ のときが最良となっている．また，平均的には，$p_{\max} = 0.04, 0.05, 0.06$ の場合が $p_{\max} = 0.0$ の場合より優れた結果となっている．

表 5 に Welded Beam 問題の実験結果を示す．なお，関数評価回数は，5,000 回である．最良値については $p_{\max} = 0.0 \sim 0.06$ の場合に最良となっており，平均値，最悪値，標準偏差について $p_{\max} = 0.05$ のときが最良となっている．また，平均的には，$p_{\max} = 0.02 \sim 0.08$ の場合が $p_{\max} = 0.0$ の場合より優れた結果となっている．

表 6 に実験結果を示す．なお，関数評価回数は，50,000 回である．最良値については，$p_{\max} = 0.0 \sim 0.06$

表 4: Result of Himmerblau's problem by pPSO

$p_{\max}$	Best	Average	Worst	S.D.
0.0	-31020.7144	-30993.4947	-30887.1383	25.3406
0.02	-31022.6666	-30990.1434	-30605.8174	48.0131
0.04	**-31024.5635**	**-31002.1995**	**-30946.7769**	**13.6621**
0.05	-31020.3756	-30996.0217	-30921.1782	18.8442
0.06	-31019.6899	-30994.5667	-30871.1132	22.7951
0.08	-31017.9066	-30991.8108	-30778.5388	27.2237
0.1	-31014.9354	-30985.5904	-30922.2199	19.4825

表 5: Result of Welded Beam problem by pPSO

$p_{\max}$	Best	Average	Worst	S.D.
0.0	**1.7249**	1.7471	1.9747	0.04547
0.02	**1.7249**	1.7357	1.8911	0.02622
0.04	**1.7249**	1.7384	2.1062	0.04474
0.05	**1.7249**	**1.7345**	**1.8267**	**0.01730**
0.06	**1.7249**	1.7414	2.0117	0.04202
0.08	1.7252	1.7427	1.8937	0.02861
0.1	1.7252	1.7547	2.0536	0.04731

の場合が最良となっており，平均値，最悪値，標準偏差については $p_{\max} = 0.08$ のときが最良となっている．また，平均的には，$p_{\max} = 0.02 \sim 0.1$ の全ての場合において $p_{\max} = 0.0$ の場合より優れた結果となっている．

表 6: Result of Pressure Vessel problem by pPSO

$p_{\max}$	Best	Average	Worst	S.D.
0.0	**6059.7143**	6295.0391	6820.4101	240.300
0.02	**6059.7143**	6218.6328	6870.7874	184.573
0.04	**6059.7143**	6141.8178	6705.0514	125.594
0.05	**6059.7143**	6138.2732	6431.4942	107.444
0.06	**6059.7143**	6124.1207	6784.9034	112.027
0.08	6059.7505	**6113.3263**	**6410.0868**	**75.4191**
0.1	6059.7149	6124.2672	7742.5623	178.403

このように，$p_{\max} = 0.0$ の場合，すなわち制約逸脱度を優先する比較のみでもかなり良い解を得ているが，0.02〜0.08 程度の確率的な逸脱を許容することにより，平均的にはよりよい解をより安定的に獲得することができる．したがって，pPSO は制約逸脱度を優先する比較よりも優れた結果を得ることができる方法であることが示された．

7 あとがき

制約付き最適化問題に対する新しいアルゴリズム変換法として，確率的制約法を提案した．確率的制約法は，制約のない問題に対するアルゴリズムにおいて，比較演算子を確率的比較に置き換えることにより，そのアルゴリズムを制約付き最適化アルゴリズムに変換する方法である．本論文では，確率的制約法をパーティクルスォーム最適化に適用した確率的制約パーティクルスォーム最適化アルゴリズム pPSO を構成した．pPSO により幾つかの代表的な制約付き最適化問題を解くことにより，確率的制約法および pPSO の有効性を示した．

今後は，確率的制約法におけるパラメータである p の制御について更に考察するとともに，確率的制約法を様々なアルゴリズムに適用し，その性能を調べることを予定している．

謝辞

本研究の一部は，日本学術振興会科学研究費補助金基盤研究 (C)(課題番号 16500083, 17510139) による補助のもとで行われた．

参考文献

[1] Z.Michalewicz. Genetic Algorithms, Numerical Optimization and Constraints, *Proc. of the 6th International Conference on Genetic Algorithms*, pp.151–158, Pittsburgh, July 1995.

[2] Z.Michalewicz and G.Nazhiyath, GENOCOP III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints, *Proc. of the 2nd IEEE International Conference on Evolutionary Computation*, vol. 2, pp.647–651, Perth, Australia, 1995.

[3] Z.Michalewicz, *Genetic algorithm + data structures = evolution programs 3rd ed.*, Springer-Verlag, Berlin, 1996.

[4] J.Kennedy and R.Eberhart, Particle Swarm Optimization, *Proc. of IEEE International Conference on Neural Networks*, vol. IV, pp.1942–1948, Perth, Australia, 1995.

[5] Y.Shi and R.Eberhart, A Modified Particle Swarm Optimizer, *Proc. of IEEE International Conference on Evolutionary Computation*, pp.69–73, Anchorage, May 1998.

[6] J.Kennedy and R.C.Eberhart, *Swarm Intelligence*, Morgan Kaufmann, San Francisco, 2001.

[7] K.E.Parsopoulos and M.N.Vrahatis, Particle swarm optimization method for constrained optimization problems, in Intelligent Technologies — Theory and Application: New Trends in Intelligent Technologies, eds. P.Sincak, J.Vascak et al., pp.214–220, IOS Press, 2002.

[8] K.Deb, An efficient constraint handling method for genetic algorithms, *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2/4, pp.311–338, 2000.

[9] T.P.Runarsson and X.Yao, Stochastic ranking for constrained evolutionary optimization, *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 3, pp.284–294, Sept. 2000.

[10] T.Takahama and S.Sakai, Tuning Fuzzy Control Rules by the α Constrained Method Which Solves Constrained Nonlinear Optimization Problems, *IEICE Trans. on Information and Systems*, vol. J82-A, no. 5, pp.658–668, May 1999, in Japanese.

[11] T.Takahama and S.Sakai, Tuning Fuzzy Control Rules by the α Constrained Method Which Solves Constrained Nonlinear Optimization Problems, *Electronics and Communications in Japan*, vol. 83, no. 9, pp.1–12, 2000.

[12] T.Takahama and S.Sakai. Learning Fuzzy Control Rules by α-Constrained Simplex Method, *IEICE Trans. on Information and Systems*, vol. J83-D-I, no. 7, pp.770–779, July 2000, in Japanese.

[13] T.Takahama and S.Sakai, Learning Fuzzy Control Rules by α-Constrained Simplex Method, *System and Computers in Japan*, vol. 34, no. 6, pp.80–90, 2003.

[14] T.Takahama and S.Sakai. Constrained Optimization by α Constrained Genetic Algorithm (αGA), *IEICE Trans. on Information and Systems*, vol. J86-D-I, no. 4, pp.198–207, Apr. 2003, in Japanese.

[15] T.Takahama and S.Sakai, Constrained Optimization by α Constrained Genetic Algorithm (αGA), *Systems and Computers in Japan*, vol. 35, no. 5, pp.11–22, May 2004.

[16] 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*, Yokohama, Japan, 2004.

[17] T.Takahama and S.Sakai, Constrained Optimization by the α Constrained Particle Swarm Optimizer, *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol.9, No.3, pp.282–289, 2005.

[18] T.Takahama and S.Sakai, Constrained Optimization by Applying the α Constrained Method to the Nonlinear Simplex Method with Mutations, *IEEE Transactions on Evolutionary Computation*, Vol.9, No.5, pp.437–451, 2005.

[19] T.Takahama and S.Sakai, Constrained Optimization by ϵ Constrained Particle Swarm Optimizer with ϵ-level Control, in Soft Computing as Transdisciplinary Science and Technology, Springer Verlag, pp.1019–1029, 2005.

[20] T.Takahama and S.Sakai, 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, Lecture Notes in Artificial Intelligence 3809, to appear.

[21] E.Camponogara and S.N.Talukdar, A genetic algorithm for constrained and multiobjective optimization, *3rd Nordic Workshop on Genetic Algorithms and Their Applications*, Vaasa, Finland, pp.49–62, Aug. 1997.

[22] P.D.Surry and N.J.Radcliffe, The COMOGA method: Constrained optimisation by multiobjective genetic algorithms, *Control and Cybernetics*, vol. 26, no. 3, pp.391–412, 1997.

[23] C.A.C.Coello, "Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art," Computer methods in applied mechanics and engineering, no.191, pp.1245–1287, 2002.