数理解析研究所講究録 1457 巻 2005 年 233-240

α制約パーティクルスォームオプティマイザαPSOによる制約付き最適化

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

与えられた制約のもとで目的関数を最小にするような解を求める制約付き最適化問題，特に非線形最適化問題は，実問題に頻繁に出現する重要な最適化問題である．制約付き最適化問題の解法としては，線形計画問題の場合はシンプレックス法がよく知られており，目的関数が非線形で微分可能な場合には，逐次２次計画法，射影法，一般縮小勾配法などの最適化手法が知られている [1, 2]．しかし，目的関数が微分可能でない場合や制約条件が非凸の場合には，これらの方法を利用することは困難である．より汎用的な方法としては，制約付き最適化問題を制約のない最適化問題に変換する変換法と目的関数値のみを用いる直接探索法を組み合わせることが有効であり，多くの応用例が存在する．

変換法 [3] には，ペナルティ法，乗数法などがあるが，制約条件の境界付近に解が存在する場合が多いこと，パラメータが少なく単純で使用しやすいことなどからペナルティ法を採用することが多いが，ペナルティ係数の値が大きくなるにつれて，変換された問題の最適化が困難になるという問題，制約条件を完全に満足する実行可能解を探索することが困難であるという問題がある [4]．

一方，直接探索法については，近年確率的多点探索を行う遺伝的アルゴリズム (Genetic Algorithm, GA)[5, 6] やパーティクルスォーム最適化 (Particle Swarm Optimization, PSO)[7, 8] が広く使用されるようになってきている．GA は，生物の進化過程を模倣した最適化アルゴリズムであり，強力かつ柔軟な最適化手法として注目されている．GA を利用した制約付き最適化に関する研究も盛んに行われるようになってきており [9]，既存の方法と比較しても遜色のない結果が得られるようになってきている [10]．特に Michalewicz らは，遺伝子を実数値とする実数値 GA(real-coded GA)[6, 11] を採用した GENOCOPIII(GEnetic algorithm for Numerical Optimization of COnstrained Problems) を提案し，一般に非線形制約をも取り扱えることを示した [12, 10]．また，α 制約法 [13, 14] と遺伝的アルゴリズムを組み合わせた α 制約遺伝的アルゴリズム (αGA) も提案されており，GENOCOP 5.0 以上の性能が示されている [15]．

本研究では，α 制約法と高速の最適化アルゴリズムとして注目されている PSO を組み合わせた α 制約パーティクルスォームオプティマイザ (αPSO) を提案する．α 制約法は，制約を満足する度合いを表現する制約満足度を導入し，通常の大小関係の代わりに制約満足度を優先した大小関係である α レベル比較を定義し，通常の比較の代わりに α レベル比較を用いて探索することにより，制約のない問題に対するアルゴリズムを制約付き問題に対するアルゴリズムに変換するアルゴリズム変換法である．文献 [13] では Powell 法 [16]，文献 [14] では Nelder&Mead の Simplex 法 [17]，文献 [15] では GA を α 制約法によって制約付き問題に対するアルゴリズムに変換することにより，目的関数が微分不可能である制約付き最適化問題を効率的に解けることが示されている．α 制約法が PSO にも適用可能であることを示すことも本論文の目的である．

α 制約法を適用した αPSO では，制約を満足しないエージェントは制約を満足するように，制約を満足したエージェントは目的関数値を最適化するように行動することになる．本論文では，非線形計画問題，非凸非線形制約など様々な種類のテスト問題について，GA による制約付き最適化手法の中で有効性がよく知られている GENOCOPIII を発展させた GENOCOP 5.0[18, 19] およびそれ以上の有効性が示されている αGA と比較することにより，αPSO の有効性を示す．さらに，PSO に基づく制約付き最適化法として提案されているペナルティ関数を用いる方法 [23] および実行可能解を保持する方法 [22] などと比較し，その特徴を示す．

以下，**2.** で本論文の対象とする制約付き最適化問題を定義し，**3.** で α 制約法を説明する．**4.** で αPSO を定義し，**5.** で αPSO の性能を示す．**6.** で PSO に基づく他の方法と比較する．**7.** はまとめである．

2　制約付き最適化問題

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．

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

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，$g_j(\boldsymbol{x})\le 0$ 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_j，h_j 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．

3　α 制約法

α 制約法について簡単に説明する．

3.1　制約付き最適化問題と制約満足度

α 制約法では，制約をどの程度満足しているかを表現するために，制約満足度 $\mu(\boldsymbol{x})$ を導入する．制約満足度 $\mu(\boldsymbol{x})$ は，以下を満足する関数である．

$$
\begin{cases}
\mu(\boldsymbol{x})=1, & \text{if } g_i(\boldsymbol{x})\le 0,\ h_j(\boldsymbol{x})=0 \text{ for all } i,j \\
0\le\mu(\boldsymbol{x})<1, & \text{otherwise}
\end{cases}
\tag{2}
$$

このような制約満足度を定義する 1 つの方法として，各制約の満足度を定義し，それらを合成する方法がある．例えば，問題 (P) における各制約条件は，機械的に以下のような g_i, h_j に関する区分的線形の制約満足度関数に変換できる．ただし，$b_i, b_j(>0)$ は適当な定数である．

$$
\mu_{g_i}(\boldsymbol{x})=\begin{cases}
1, & \text{if } g_i(\boldsymbol{x})\le 0 \\
1-\frac{g_i(\boldsymbol{x})}{b_i}, & \text{if } 0\le g_i(\boldsymbol{x})\le b_i \\
0, & \text{otherwise}
\end{cases}
\qquad
\mu_{h_j}(\boldsymbol{x})=\begin{cases}
1-\frac{|h_j(\boldsymbol{x})|}{b_j}, & \text{if } |h_j(\boldsymbol{x})|\le b_j \\
0, & \text{otherwise}
\end{cases}
\tag{3}
$$

この各制約満足度から制約全体の満足度 $\mu(\boldsymbol{x})$ を求めるために，各制約満足度を結合する．結合演算としては，以下のような min 演算が考えられる．

$$
\text{min 演算}\quad \mu(\boldsymbol{x})=\min_{i,j}\{\mu_{g_i}(\boldsymbol{x}),\mu_{h_j}(\boldsymbol{x})\}
\tag{4}
$$

3.2　α レベル比較

関数値と制約満足度の組 (f,μ) の集合上において，制約満足度が α 以上の場合は目的関数値の大小関係を優先し，それ以外の場合は制約満足度の大小関係を優先する比較である α レベル比較を定義する．

点 $\boldsymbol{x}_1,\boldsymbol{x}_2$ における関数値を f_1,f_2，制約満足度を μ_1,μ_2 とすると，通常の大小関係である $\le$ に対応する関数値と制約満足度の組 (f_i,μ_i) 間の大小関係である α レベル比較 $<_\alpha$ および $\le_\alpha$ $(0\le\alpha\le 1)$ は以下のようになる．

$$
(f_1,\mu_1)<_\alpha(f_2,\mu_2)\Leftrightarrow\begin{cases}
f_1<f_2, \text{ if } \mu_1,\mu_2\ge\alpha \\
f_1<f_2, \text{ if } \mu_1=\mu_2 \\
\mu_1>\mu_2, \text{otherwise}
\end{cases}
\qquad
(f_1,\mu_1)\le_\alpha(f_2,\mu_2)\Leftrightarrow\begin{cases}
f_1\le f_2, \text{ if } \mu_1,\mu_2\ge\alpha \\
f_1\le f_2, \text{ if } \mu_1=\mu_2 \\
\mu_1>\mu_2, \text{otherwise}
\end{cases}
\tag{5}
$$

なお，$<_0$，$\le_0$ は関数値のみの比較と一致し，$<_1$，$\le_1$ は制約満足度を優先する辞書式比較と一致する．

3.3 α 制約法の性質

α 制約法は，制約付き最適化問題を直接探索法で解く際に，通常の比較の代わりに α レベル比較を用いる方法である．通常の大小比較を α レベル比較に置き換えた最適化問題 $(\mathrm{P}_{\leq_\alpha})$，すなわち，$\alpha$ 制約法による最適化問題は以下のように定義できる．但し，$\mathrm{minimize}_{\leq_\alpha}$ は $\leq_\alpha$ の意味での最小化である．

$$(\mathrm{P}_{\leq_\alpha}) \quad \mathrm{minimize}_{\leq_\alpha} \quad f(\boldsymbol{x}) \tag{6}$$

ここで，問題 (P) の制約条件を $\mu(\boldsymbol{x}) \geq \alpha$ に緩和した問題 (P^α) を以下のように定義する．なお，(P^1) は問題 (P) と等価である．

$$\begin{array}{lll} (\mathrm{P}^\alpha) & \text{minimize} & f(\boldsymbol{x}) \\ & \text{subject to} & \mu(\boldsymbol{x}) \geq \alpha \end{array} \tag{7}$$

問題 (P^α) と問題 $(\mathrm{P}_{\leq_\alpha})$，および問題 (P) に関して以下の定理が成り立つ [13]．

定理 1 問題 *(P^1)* に最適解が存在するならば，問題 *($P_{\leq_\alpha}$)* の最適解は問題 *(P^α)* の最適解である．

定理 2 問題 *(P)* に最適解が存在するならば，問題 *($P_{\leq_1}$)* の最適解は，問題 *(P)* の最適解である．

定理 3 $\{\alpha_n\}$ を強い意味で単調増加し *1* に収束する点列とする．$f(\boldsymbol{x})$, $\mu(\boldsymbol{x})$ を連続関数とし，問題 *(P^1)* の最適解 $\boldsymbol{x}^*$ の存在と任意の α_n に対する問題 *($P_{\leq_{\alpha_n}}$)* の最適解 $\hat{\boldsymbol{x}}_n$ の存在を仮定する．このとき，点列 $\{\hat{\boldsymbol{x}}_n\}$ の任意の集積点は問題 *(P^1)* の最適解である．

定理 1, 2 は，α レベル比較を行うことにより，制約付き問題がそれと等価な制約のない問題に変換されることを示している．したがって，既存の制約のない最適化手法に α レベル比較を導入することにより，制約付きの問題を解くことが可能となる．定理 3 は，ペナルティ法においてペナルティ係数を ∞ まで増加させて行くのと同様に，α を 1 まで増加させながら最適化を行っても，最適解が得られることを示している．

4 α 制約パーティクルスォームオプティマイザ αPSO

α 制約法は，目的関数値の大小関係のみに基づく最適化アルゴリズムと組み合わせることができる．本研究では，目的関数の大小関係のみに依存するアルゴリズムであるパーティクルスォーム最適化 (PSO) に α 制約法を適用する．

4.1 パーティクルスォーム最適化 PSO

まず，PSO について簡単に説明する．ある種の動物は敵から身を守ったり，餌や仲間を探すために，個別に行動するのではなく，群れ (グループ) を作って行動する．鳥の群れ (flock of birds)，魚の群れ (school of fish) あるいは昆虫の群れ (swarm of insects) などがこの例である．グループ中には，各動物（以下エージェントと呼ぶ）が餌などを探すときにグループ内の他のエージェントと情報を共有するという社会的な関係が存在すると考えられる．すなわち，各エージェントは各自の経験だけではなく，他の全てのエージェントの経験を共有し，その情報から利益を受けると仮定する．PSO は，このような仮定に基づく各エージェントの単純な行動ルールによってグループとしての行動を実現し，最適化に応用しようというものである．

PSO では，各エージェントが解候補に対応し，エージェントのグループにより解を探索する．また，エージェントの行動は確定的ではなく，ランダム性が導入されている．したがって，PSO は GA と同様に確率的多点探索法の一つである．GA における個体および集団は，PSO におけるパーティクルおよびスォームに対応する．また，PSO は単純な原理に基づいており，基本的な数学的演算のみで実現できるため，効率的な計算が可能となり，高速に実行でき，必要なメモリも少ない方法であるとされている．なお，本論文ではパーティクルをエージェントと呼ぶことにする．

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．

$$pbest_i = \min_{\tau=0,1,\cdots,t} f(\boldsymbol{x}_i^\tau), \quad \boldsymbol{x}_i^* = \arg\min_{\tau=0,1,\cdots,t} f(\boldsymbol{x}_i^\tau) \tag{8}$$

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．

$$gbest = \min_i pbest_i, \quad \boldsymbol{x}_G^* = \arg\min_i f(\boldsymbol{x}_i^*) \tag{9}$$

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．

$$\boldsymbol{v}_i^{t+1} = w\boldsymbol{v}_i^t + c_1\, rand \times (\boldsymbol{x}_i^* - \boldsymbol{x}_i^t) + c_2\, rand \times (\boldsymbol{x}_G^* - \boldsymbol{x}_i^t) \tag{10}$$

ただし，w は慣性重み (inertia weight)，$rand$ は区間 $[0,1]$ の一様乱数である．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 (10) 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．

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \tag{11}$$

PSO において w は解への収束を制御する重要なパラメータである．w が大きい場合は新しい領域を探索する，すなわち大域的探索を行う傾向が強くなり，小さい場合には現在の位置の周辺領域を探索する，すなわち局所的探索を行う傾向が強くなる．w を適切に調整することにより，大域的探索と局所探索のバランスをとることができる．初期には探索空間を網羅的に調べるために大域探索を強くし，精度の高い解を得るために次第に局所探索を強くする，すなわち初期には w を大きくとり，次第に減少させるという方法が提案されている．また，速度が過大にならないように，最大速度を与え，それを超えないように速度を制限することが多い．本研究でも，w を減少させ，最大速度を与えるという方法を採用する．

4.2 α 制約パーティクルスォームオプティマイザ αPSO

αPSO のアルゴリズムを以下に示す．αPSO のアルゴリズムは，PSO における比較を α レベル比較で置換し，α レベルの制御を追加したものとなっている．

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，各次元毎の最大速度 $v_k^{\max}$ 以下，すなわち，区間 $[-v_k^{\max}, v_k^{\max}]$ の一様乱数とする．

2. 最良エージェントの決定：α レベル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. α レベルの決定：通常は $\alpha = 1$ として探索すればよいが，等式制約のように実行可能領域が非常に狭い場合には，実行可能領域が狭すぎて有効な最適化が行えない．このような場合には α レベルを制御し，初期には制約条件を緩和して目的関数の最適化を優先し，次第に制約条件を厳しくする，すなわち α を 1 に近づけてゆく必要がある (**4.3** 参照)．

5. エージェントの更新：各エージェント i について，式 (10),(11) により移動速度および位置を更新する．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．

6. (3) へ戻る．

αPSO のアルゴリズムを以下に C 言語風に記述する.

```
αPSO()
{
  Initialize P(0);                      ······· (1)
  α=α(0);
  x*_G=arg min_{<α} f(x), x in P(0);   ······· (2)
  for(t=1;t≤T;t++) {                    ······· (3)
    α=α(t);                             ······· (4)
    w=w^0 + (w^T − w^0) t/T;
    for(each agent i in P(t)) {         ······· (5)
      v_i=wv_i+c_1 rand(x*_i-x_i)+c_2 rand(x*_G-x_i);
      x_i=x_i+v_i;
      if(f(x_i)<_α f(x*_i)) {
        x*_i=x_i;
        if(f(x_i)<_α f(x*_G)) x*_G=x_i;
} } } }
```

ただし, $\alpha(t)$ は反復回数 t における α レベルである. 慣性重み w は, w^0 から w^T まで変化させる [20, 21].

4.3 α レベルの制御

一般に α 制約法を多点探索法と組み合わせたとき, α レベルを満足する探索点 (エージェント) が少なくなると, 目的関数値を無視して制約条件を満足する方向に探索する傾向が強くなる. 制約を満足する領域が広い問題では, これにより探索点集合が急速に実行可能領域に近づき, 実行可能領域を集中的に探索することができるため, 効率の良い探索が可能となる. しかし, 特に等式制約のように実行可能領域が極端に狭い問題では, 実行可能領域の発見が困難なため, 制約満足度の高い探索点の付近に他の探索点が集中してゆき, 探索領域が狭くなってしまい, 目的関数値の最適化が困難になる. これを避けるためには, α レベルを制御し, 問題の制約を緩和する必要がある.

本論文では, 文献 [15] の制御方法を採用した. すなわち, 以下のように初期値 $\alpha(0)$ を初期グループの制約満足度の平均値と最大値の中間とし, 最大反復回数 T の半分以降は常に 1 となる反復回数 t の 2 次式に基づく制御とした.

$$\alpha(0) = \frac{1}{2}(\max_{\boldsymbol{x}_i} \mu(\boldsymbol{x}_i) + \frac{1}{N}\sum_{i=1}^{N} \mu(\boldsymbol{x}_i)), \quad \alpha(t) = \begin{cases} 1-(1-\alpha(0))(1-\frac{2t}{T})^2, & 0 < t < \frac{T}{2}, \\ 1, & t \geq \frac{T}{2}. \end{cases} \tag{12}$$

ただし, N はエージェント数である.

5 制約付き非線形計画問題

本論文では, αPSO により幾つかの最適化問題を対象に最適化を行い, その結果について GENOCOP 5.0 および αGA と比較検討する.

5.1 テスト問題と実験条件

表 1 に示した Michalewicz が提案した 5 つの問題 $G_1 \sim G_5$[10], および坂和らが提案した 1 つの最適化問題 S_1[4] を対象とする (文献 [15] 参照). G_1 は線形制約, G_2 は線形目的関数, G_4 は等式制約の問題であり, S_1 は非凸非線形制約問題である.

αPSO の設定では, 全ての問題に対して同一の設定を用いた. α 制約法に関する設定については, 制約満足度の結合演算を min 演算, 制約満足度のパラメータを $b_i = b_j = 10000$ とした. 等式制約を含む G_4 のみ式 (12) による α の制御を行い, その他の問題については $\alpha = 1$ に固定した. PSO に関する設定については, エージェント数 $N = 70$, 最大速度を各問題における変数の上下限制約の幅 $v_k^{\max} = u_k - l_k$, 慣性重み初期値 $w^0 = 1$, 最終値 $w^T = 0.2$, 重みパラメータ $c_1 = c_2 = 2$ とした. 最大反復回数 $T = 5000$ を一試行とし, 各問題について 100 試行の実験を行った. なお, GENECOP5.0 と αGA の実験結果は文献 [15] に基づいている. PSO におけるエージェント数および最大反復回数は, この実験条件に準じている.

表 1: テスト問題

$G_1(\boldsymbol{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i,$

subject to
$2(x_1+x_2)+x_{10}+x_{11} \le 10,$
$2(x_1+x_3)+x_{10}+x_{12} \le 10,$
$2(x_2+x_3)+x_{11}+x_{12} \le 10,$
$-8x_1+x_{10} \le 0,\ -8x_2+x_{11} \le 0,$
$-8x_3+x_{12} \le 0,\ -2x_4-x_5+x_{10} \le 0,$
$-2x_6-x_7+x_{11} \le 0,\ -2x_8-x_9+x_{12} \le 0,$
$0 \le x_i \le 1, i=1,\dots,9,$
$0 \le x_i \le 100, i=10,11,12,\ 0 \le x_{13} \le 1.$

$G_2(\boldsymbol{x}) = x_1+x_2+x_3,$

subject to
$1-0.0025(x_4+x_6) \ge 0,$
$1-0.0025(x_5+x_7-x_4) \ge 0,$
$1-0.01(x_8-x_5) \ge 0,$
$x_1x_6-833.33252x_4-100x_1+83333.333 \ge 0,$
$x_2x_7-1250x_5-x_2x_4+1250x_4 \ge 0,$
$x_3x_8-1250000-x_3x_5+2500x_5 \ge 0,$
$100 \le x_1 \le 10000,\ 1000 \le x_i \le 10000, i=2,3,$
$10 \le x_i \le 1000, i=4,\cdots,8.$

$$G_3(\boldsymbol{x}) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2 +10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7,$$

subject to
$127-2x_1^2-3x_2^4-x_3-4x_4^2-5x_5 \ge 0,$
$282-7x_1-3x_2-10x_3^2-x_4+x_5 \ge 0,$
$196-23x_1-x_2^2-6x_6^2+8x_7 \ge 0,$
$-4x_1^2-x_2^2+3x_1x_2-2x_3^2-5x_6+11x_7 \ge 0,$
$-10 \le x_i \le 10, i=1,\dots,7.$

$G_4(\boldsymbol{x}) = e^{x_1x_2x_3x_4x_5},$

subject to
$x_1^2+x_2^2+x_3^2+x_4^2+x_5^2=10,$
$x_2x_3-5x_4x_5=0,\ x_1^3+x_2^3=-1,$
$-2.3 \le x_i \le 2.3, i=1,2,\ -3.2 \le x_i \le 3.2, i=3,4,5.$

$$G_5(\boldsymbol{x}) = x_1^2+x_2^2+x_1x_2-14x_1-16x_2+(x_3-10)^2 +4(x_4-5)^2+(x_5-3)^2+2(x_6-1)^2+5x_7^2 +7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2+45$$

subject to
$105-4x_1-5x_2+3x_7-9x_8 \ge 0,$
$-10x_1+8x_2+17x_7-2x_8 \ge 0,$
$8x_1-2x_2-5x_9+2x_{10}+12 \ge 0,$
$-3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4+120 \ge 0,$
$-5x_1^2-8x_2-(x_3-6)^2+2x_4+40 \ge 0,$
$-x_1^2-2(x_2-2)^2+2x_1x_2-14x_5+6x_6 \ge 0,$
$-0.5(x_1-8)^2-2(x_2-4)^2-3x_5^2+x_6+30 \ge 0,$
$3x_1-6x_2-12(x_9-8)^2+7x_{10} \ge 0,$
$-10 \le x_i \le 10, i=1,\dots,10.$

$$S_1(\boldsymbol{x}) = x_1^3+(x_2-5)^2+3(x_3-9)^2-12x_3+2x_4^3 +4x_5^2+(x_6-5)^2-6x_7^2+3(x_7-2)x_8^2 -x_9x_{10}+4x_9^3+5x_1x_3-3x_1x_7+2x_8x_7,$$

subject to
$-3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4 -2x_5x_6x_8+120 \ge 0,$
$-5x_1^2-8x_2-(x_3-6)^2+2x_4+40 \ge 0,$
$-x_1^2-2(x_2-2)^2+2x_1x_2-14x_5-6x_5x_6 \ge 0,$
$-0.5(x_1-8)^2-2(x_2-4)^2-3x_5^2+x_5x_8+30 \ge 0,$
$3x_1-6x_2-12(x_9-8)^2+7x_{10} \ge 0,$
$4x_1+5x_2-3x_7+9x_8 \le 105,$
$10x_1-8x_2-17x_7+2x_8 \le 0,$
$-8x_1+2x_2+5x_9-2x_{10} \le 12,$
$-5 \le x_i \le 10, i=1,\dots,10.$

5.2 実験結果および評価

問題 $G_1 \sim G_5$ に対する実験結果を表 2 に示す．ただし，best, average, worst, σ はそれぞれ 100 回の試行における最良エージェント (個体) の目的関数値の最良値，平均値，最悪値，標準偏差であり，CPU は UltraSPARCIII(750MHz) を使用したときの 1 試行当たりの平均実行時間である．#gen, #func, #const は，それぞれ最良エージェントが得られた平均反復回数 (世代数)，その回数までの目的関数の平均評価回数，制約条件の平均評価回数である．等式制約を含む G_4 では，最良エージェントの制約条件からの逸脱の平均値を const_err に示した．

αPSO は全ての問題について，最良値，平均値，最悪値，標準偏差が GENOCOP 5.0 と同じあるいはより優れている．また，G_2, G_3, G_4, S_1 については，S_1 の最悪値を除いて，最良値，平均値，最悪値，標準偏差が αGA よりも優れている．特に等式制約を持つ G_4 については，αPSO では全ての試行において 0.05 台の解を発見し，αGA と比較しても制約からの逸脱が非常に小さい解を発見している．G_1 では，αGA が平均値，最悪値において αPSO より優れているがその差は僅かである．G_5 では，αGA が平均値，最悪値において αPSO より優れているが，最良値については，αGA が発見できなかった 24.3 台の解を発見している．したがって，αPSO は明らかに GENOCOP 5.0 よりも優れており，αGA と同等あるいはより優れた探索能力を持つアルゴリズムといえる．さらに，各問題の最適値と αPSO の平均値の差はわずかであり，局所解に陥りにくいアルゴリズムであることが分かった．

実行時間，目的関数および制約条件の評価回数についても，全ての問題において αPSO は最も優れている．αPSO の実行速度は，αGA の 3〜4 倍程度，GENOCOP 5.0 の 10 倍以上となっている．目的関数の評価回数は，αGA の 8〜60%程度 GENOCOP 5.0 の 4〜30%程度であり，制約条件の評価回数も αGA の 11〜18%程度 GENOCOP 5.0 の 1〜4%程度とかなり少なくなっている．αGA では境界突然変異などの突然変異を高い頻度で用いており，その際，個体が実行可能でない場合には直線探索により実行可能領域を探すという処理を行うため，制約条件の評価回数が多くなっている．αPSO では単純に処理すれば，目的関数の評価回数は制約条件の評価回数と一致するが，α 制約法では α レベルを満足しない解を比較する場合には制約条件だけで大小関係を判断できるため，目的関数の評価を省略することができる．αPSO では新しい探索点を過去の最良値とグループの最良値のみと比較するため，この省略がかなり有効に働いており，実行可能領域が見つかるまでは制約条件の評価だけを行っていることが分かる．なお，αGA ではランク選択を行うため，集団中の全ての個体と比較する必要があり，この省略の効果が αPSO ほどは高くない．

以上のことから，αPSO は非常に高速であり，局所解に陥りにくい優れたアルゴリズムであるといえる．

表 2: 実験結果

f	Item	αPSO	αGA	GCOP5.0
G_1	best	-15.000	-15.000	-15.000
	average	-15.000	-15.000	-11.523
	worst	-14.999	-15.000	-3.442
	σ	0.000	0.000	4.447
	CPU(s)	2.451	7.120	24.810
	#gen	4933.0	3248.4	3369.9
	#func	14,957	141,338	235,964
	#const	345,351	1,984,329	8,540,452
G_2	best	7049.412	7053.951	7087.337
	average	7430.288	7514.233	8114.309
	worst	8411.988	10855.844	11107.814
	σ	231.000	661.926	966.195
	CPU(s)	1.689	6.631	19.470
	#gen	4966.2	4135.4	4762.6
	#func	18,887	180,916	333,448
	#const	347,668	3,234,494	12,398,935
G_3	best	680.631	680.646	680.634
	average	680.641	680.687	680.750
	worst	680.667	680.773	683.255
	σ	0.007	0.024	0.289
	CPU(s)	1.571	7.165	18.590
	#gen	4930.2	2715.4	3591.7
	#func	72,089	119,097	251,490
	#const	345,155	2,584,109	9,305,135
G_4	best	0.05395	0.05396	0.05583
	average	0.05437	0.19049	1.40557
	worst	0.05641	1.00802	18.68213
	σ	0.00051	0.21249	2.53356
	const_err	2.220e-16	1.685e-5	8.836e-4
	CPU(s)	1.368	4.451	28.240
	#gen	3399.8	3745.8	4587.0
	#func	13,288	165,408	321,162
	#const	238,025	1,442,838	21,883,244
G_5	best	24.311	24.401	24.529
	average	25.173	24.542	28.362
	worst	27.787	24.844	48.058
	σ	0.690	0.090	3.926
	CPU(s)	2.084	8.525	27.530
	#gen	4980.9	3419.7	4663.7
	#func	32,976	149,168	326,532
	#const	348,701	2,811,833	12,914,562
S_1	best	-216.656	-216.654	-216.450
	average	-201.911	-187.319	-78.708
	worst	9.344	7.217	473.098
	σ	46.565	64.434	166.126
	CPU(s)	2.081	8.451	24.560
	#gen	4996.1	3983.4	4415.6
	#func	25,232	173,762	309,161
	#const	349,767	3,224,557	11,368,803

6 むすび

パーティクルスォーム最適化に α 制約法を適用した αPSO を提案した．様々なタイプの制約付き最適化問題に αPSO を適用し，全ての問題で最適値に近い解が得られることを確認し，αPSO が精度の高い安定したアルゴリズムであることを示した．さらに，遺伝的アルゴリズムを利用して制約付き最適化問題を解く方法の中で有効性がよく知られている GENOCOP 5.0 および αGA と比較することにより，αPSO がこれらの方法と同等あるいはそれ以上の解を非常に高速に求めることのできる優れた方法であることを示した．さらに，PSO に基づく他の制約付き最適化方法と比較することにより，αPSO がそれらの方法と比較しても優れていることを示した．

しかし，αPSO のパラメータ設定に関してはまだ改良の余地があると思われる．例えば，エージェント数 N や w の適切な値の選択方法，αPSO に適した α レベルの制御についてはさらなる検討が必要である．

今後は，αPSO の性質についてさらに調査するとともに，様々な実問題に対して αPSO を適用していく予定である．

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)(2)(課題番号 14580498, 16500083) による補助のもとで行われた．

参考文献

[1] 今野浩, 山下浩, "非線形計画法," 日科技連, 1978.

[2] 坂和正敏, "非線形システムの最適化," 森北出版, 1986.

[3] A.V. Fiacco, G.P. McCormick, "Nonlinear programming : Sequential unconstrained minimization techniques," Society for Industrial and Applied Mathematics, 1990.

[4] 坂和正敏, 矢内克裕, "非凸非線形計画問題に対する浮動小数点型遺伝的アルゴリズム: 改良型 GENOCOPIII," 信学論 (A), vol.J81-A, no.1, pp.90–97, Jan. 1998.

[5] D.E.Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989.

[6] 坂和正敏, 田中雅博, 遺伝的アルゴリズム, 朝倉書店, 1995.

[7] J.Kennedy and R.C.Eberhart, "Particle swarm optimization," Proc. of IEEE International Conference on Neural Networks, vol.IV, pp.1942–1948, Perth, Australia, 1995.

[8] J.Kennedy and R.C.Eberhart, "Swarm intelligence," Morgan Kaufmann, San Francisco, 2001.

[9] Z.Michalewicz, "A survey of constraint handling techniques in evolutionary computation methods," Proc. of the 4th Annual Conference on Evolutionary Programming, pp. 135–155, San Diego, USA, Mar. 1995.

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

[11] 小野功, 山村雅幸, 喜多一, "実数値 GA とその応用," 人工知能学会誌, vol.15, no.2, pp.259–266, Mar. 2000.

[12] Z.Michalewicz and M.Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," Evolutionary Computation, vol.4, no.1, pp.1–32, 1996.

[13] 高濱徹行, 阪井節子, "制約付き非線形最適化手法 α 制約法によるファジー制御ルールの最適化," 信学論 (A), vol.J82-A, no.5, pp.658–668, May 1999.

[14] 高濱徹行, 阪井節子, "α 制約 Simplex 法によるファジー制御ルールの学習," 信学論 (D-I), vol.J83-D-I, no.7, pp.770–779, July 2000.

[15] 高濱徹行, 阪井節子, "α 制約遺伝的アルゴリズム αGA による制約付き最適化," 信学論 (D-I), vol.J86-D-I, no.4, pp.198–207, Apr. 2003.

[16] M.J.D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," Computer J., vol.7, pp.155–162, 1964.

[17] J.A.Nelder and R.Mead: "A simplex method for function minimization," J. Computer, vol.7, pp. 308–313, 1965.

[18] S.Koziel and Z.Michalewicz, "A Decoder-based evolutionary algorithm for constrained parameter optimization problems," Proc. of the 5th Parallel Problem Solving from Nature, Lecture Notes in Computer Science, vol.1498, pp.231–240, Amsterdam, Netherlands, Sep. 1998.

[19] S.Koziel and Z.Michalewicz, "Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization," Evolutionary Computation, vol.7, no.1, pp.19–44, 1999.

[20] Y.Shi, R.C.Eberhart, "Parameter selection in particle swarm optimization," Evolutionary Programming VII, pp.591–600, 1998.

[21] Y.Shi, "http://www.engr.iupui.edu/~eberhart/web/ PSObook.html"

[22] X.Hu and R.C.Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," Proc. of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002, Orlando, USA, 2002.

[23] 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, vol.76 of Frontiers in Artificial Intelligence and Applications, eds. P.Sincak, J.Vascak et al., pp.214–220, IOS Press, 2002.

[24] T.Ray, K.M.and Liew, "A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problem," Proc. of IEEE Congress on Evolutionary Computation 2001, Seoul, Korea, pp.75–80, 2001.

[25] G.Coath and S.K.Halgamuge, "A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems," Proc. of IEEE Congress on Evolutionary Computation 2003, Canbella, Australia. pp.2419–2425, 2003.