計算知能の逐次近似多目的最適化への応用

中山　弘隆，　尹　禮分

実問題の多くはいくつかの視点から評価を行い，意思決定することが多く，数理計画の枠組みの中では多目的最適化問題として定式化される．また，工学設計のように関数の評価のためには実実験やシミュレーションによってはじめて関数の値が分かり，しかもこのような実験やシミュレーションは通常，多大のコストがかかるため，なるべく少ない関数評価回数で解を決定したいことが多くある．本解説ではなるべく少ない実験やシミュレーションによって近似モデル（メタモデル）を作りながら，最適解を見いだそうとする逐次近似多目的最適化について計算知能（特に機械学習）を用いた方法を紹介する．

キーワード：計算知能，多目的最適化，逐次近似最適化，メタモデリング，能動学習

1. はじめに

近年のめざましい最適化技法およびそのソフトウェアの発展により，多くの実問題に最適化技術が適用されてきている．しかし，工学設計問題などでは目的関数や制約関数が設計変数の陽な関数として得られず，構造解析，熱解析，流体解析，振動解析などの数値シミュレーションや実機での実験を行うことで，はじめてそれらの関数値が得られ，しかもこれらのシミュレーションや実験には大きなコストがかかることがしばしばある．このようなとき，従来の数理計画型の最適化手法はそのままでは適用できない．このように十分な精度の数理計画モデルが前もって得られないとき，できる限り少ないサンプルで近似モデルを作成し，得られた関数に対し最適化を行い，必要があれば新たなサンプルを追加して近似モデルの精度を上げるという繰り返しによって解を求めるという逐次近似最適化が注目されている [9, 11]．このような近似モデルの作成はメタモデリングと呼ばれている．メタモデリングにおいて重要なことはいかに少ないサンプルで精度良い解を導出できるかである．本解説ではいくつかの計算知能の技法がメタモデリングや得られた近似モデルに対する最適解の導出において効果的に用いられることを示す．

2. メタモデリング

簡単化のために，$\boldsymbol{x}^* = \arg\ \min_{\boldsymbol{x}\in X}\ f(\boldsymbol{x})$ となるような設計変数 $\boldsymbol{x}^*$ を見つける問題を考える．$X \subset \mathbb{R}^n$ は実行可能（制約）集合であり，制約関数 $g_i(\boldsymbol{x}) \leqq 0\ (i = 1, \ldots, m)$ によって与えられる場合もあるが，ここでは目的関数 f のモデル化に注意を払うことにする．我々の想定する状況では f を前もって同定することが困難であり，したがっていくつかのサンプルから近似モデル $\hat{f}$ を作成し，得られた $\hat{f}$ に対し最適解を求め，必要であればサンプルを追加し近似モデルの精度を上げて，また最適化を行うという逐次近似最適化について少し詳しく述べることにする．

メタモデリングは機械学習においては適切な追加学習点を求め，追加学習していく能動学習となる．ここにおいて重要なことはいかに少ないサンプルで近似モデルの精度を上げ，精度の高い近似最適解を得させるかである．よく用いられるメタモデリング手法を以下に簡単に述べる．

2.1 線形回帰

次式で与えられるモデルを考える：

$$\hat{f}(\boldsymbol{x}) = \sum_{j=1}^{m} w_j h_j(\boldsymbol{x})$$

基底関数 h_j としては，多項式，シグモイド関数，スプライン関数等，種々のものが考えられているが，なかでも放射基底関数 (radial basis function: RBF) を用いた RBF ネットワークは実際の応用でも良く用いられている．代表的な RBF はガウス関数である．ここで精度良いモデルを作るために適切な $w_j\ (j = 1, \ldots, m)$ を定めることが学習となる．学習サンプル $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ に対し，

なかやま　ひろたか
甲南大学 知能情報学部
〒 658–8501 神戸市東灘区岡本 8–9–1
ゆん　いえぶん
関西大学　環境都市工学部
〒 564–8680 大阪府吹田市山手町 3–3–35

$$y_i = \hat{f}(\boldsymbol{x}_i, \boldsymbol{w}) + \varepsilon_i, \quad i = 1, \ldots, \ell \tag{1}$$

とし，観測誤差 ε_i は $N(0, \sigma^2)$ に従い，互いに独立とする．このとき，通常よく用いられる最小 2 乗法と尤度最大化は同等となる．すなわち，$\sum_{i=1}^{\ell}(y_i - \hat{f}(\boldsymbol{x}_i))^2$ を最小化する解 $\hat{\boldsymbol{w}}$ は

$$\hat{\boldsymbol{w}} = \left(H^T H\right)^{-1} H^T \boldsymbol{y}$$

で与えられる．ただし，

$$H = \begin{bmatrix} h_1(\boldsymbol{x}_1) & \cdots & h_m(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ h_1(\boldsymbol{x}_\ell) & \cdots & h_m(\boldsymbol{x}_\ell) \end{bmatrix}$$

である．また，

$$\mathrm{Cov}(\hat{\boldsymbol{w}}) = \left(H^T H\right)^{-1} \sigma^2 \tag{2}$$

となる．

実際には，機械学習においては素子（各 h_j のこと）の過剰反応を抑えた

$$\sum_{i=1}^{\ell} \left(y_i - \hat{f}(\boldsymbol{x}_i)\right)^2 + \lambda \sum_{j=1}^{m} w_j^2 \tag{3}$$

を最小化することが多い．このとき，最適解において $H^T H$ の代わりに $H^T H + \lambda I$ とすればよい．（これは正則化あるいはリッジ回帰といわれる）．ただし，I は $m \times m$ 単位行列である．

2.2 サポートベクター回帰

近年，最も有力な機械学習の方法の一つとしてサポートベクターマシン (support vector machine: SVM) [2] がある．SVM を回帰に応用したのがサポートベクター回帰 (support vector regression: SVR) である．SVR では非線形写像 $\boldsymbol{\Phi}$ による像空間（特徴空間）での線形回帰

$$f(\boldsymbol{z}) = \boldsymbol{w}^T \boldsymbol{z} + b \tag{4}$$

をまず考える．ここで，Vapnik は ε-不感損失関数 (insensitive loss function)

$$L^{\varepsilon}(\boldsymbol{z}, y, f) = |y - f(\boldsymbol{z})|_{\varepsilon} = \max\left\{0,\ |y - f(\boldsymbol{z})| - \varepsilon\right\}$$

を導入し，これを最小化することによって最適な $\hat{\boldsymbol{w}}$ と $\hat{b}$ を求める方法を提案した．ε-不感損失関数の導入は，回帰関数に利いてくるデータ（サポートベクター[1]）が疎 (sparse) になることを可能にしている．

Schölkopf ら [13] は ε の値が自動的に出てくる ν-SVR を提案しているが，工学設計の問題などでは ε の値は許容誤差として設計者の方から指定したいことも多い．このような観点から筆者等によって (μ-SVR) が提案された:

(μ-SVR)

$$\begin{aligned} \min_{\boldsymbol{w}, b, \xi, \xi'} \quad & \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w} + \mu(\xi + \xi') \\ \text{s.t.} \quad & \left(\boldsymbol{w}^T \boldsymbol{z}_i + b\right) - y_i \leqq \varepsilon + \xi,\ \ i = 1, \ldots, \ell \\ & y_i - \left(\boldsymbol{w}^T \boldsymbol{z}_i + b\right) \leqq \varepsilon + \xi',\ \ i = 1, \ldots, \ell \\ & \varepsilon,\ \ \xi,\ \ \xi' \geqq 0 \end{aligned}$$

(μ-SVR) の双対問題は次のようになる．

(μ-SVR$_D$)

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}'} \quad & -\frac{1}{2} \sum_{i,j=1}^{\ell} (\alpha_i' - \alpha_i)(\alpha_j' - \alpha_j)\, \boldsymbol{z}_i^T \boldsymbol{z}_j \\ & + \sum_{i=1}^{\ell} (\alpha_i' - \alpha_i)\, y_i - \varepsilon \sum_{i=1}^{\ell} (\alpha_i' + \alpha_i) \\ \text{s.t.} \quad & \sum_{i=1}^{\ell} (\alpha_i' - \alpha_i) = 0 \\ & \sum_{i=1}^{\ell} \alpha_i' \leqq \mu, \quad \sum_{i=1}^{\ell} \alpha_i \leqq \mu \\ & \alpha_i' \geqq 0,\ \ \alpha_i \geqq 0,\ \ i = 1, \ldots, \ell \end{aligned}$$

ここで，μ は通常十分大きい値にすることが推奨されている [11].

実際には双対問題を用いて解を求めることが多いが，目的関数にある内積 $\boldsymbol{z}_i^T \boldsymbol{z}_j$ は，$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\Phi}(\boldsymbol{x})^T \boldsymbol{\Phi}(\boldsymbol{x}')$ となるカーネル関数を用いて計算される（カーネルトリック）．代表的なカーネルとして，ガウスカーネル

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2r^2}\right) \tag{5}$$

がよく用いられる．ガウスカーネルを用いる場合は，RBF ネットワークによる結果と同じように見えるが，基底をおく場所がサポートベクターとして自動的に出てくるところが大きく異なる．

2.3 クリギング

多くの工学設計ではコンピュータ実験によって $\boldsymbol{x}_i$ における評価関数の値 y_i が得られるが，コンピュータ実験による実験値 y_i には観測誤差は生じない．したがって，このときは回帰における誤差 ε はモデル誤差と考えるのが妥当で，それらのモデル誤差 $\varepsilon(\boldsymbol{x}_i)$ は他の $\boldsymbol{x}_j$ での誤差の影響を受けると考えられる．したがって，

$$y(\boldsymbol{x}) = \mu + \varepsilon(\boldsymbol{x}) \tag{6}$$

[1] SVR モデルにおけるラグランジュー乗数 α_i が正になるような点 $\boldsymbol{z}_i$ または $\boldsymbol{x}_i$ をサポートベクターと呼ぶ．

で表される確率過程モデルを考え[2]，μ は確率過程の平均で，$\varepsilon(\boldsymbol{x})$ は各 $\boldsymbol{x}$ において正規分布 $N(0,\sigma^2)$ に従うが，他の $\boldsymbol{x}'$ での誤差と相関 $\mathrm{Corr}(\varepsilon(\boldsymbol{x}),\varepsilon(\boldsymbol{x}'))$ があるとする．たとえば

$$\mathrm{Corr}(\varepsilon(\boldsymbol{x}),\varepsilon(\boldsymbol{x}')) = \exp\left(-\sum_{i=1}^{n}\frac{|x_i - x_i'|^{p_i}}{r_i^2}\right)$$

とすることが多い．クリギングでは相関のパラメータ r_i と $0 < p_i \leqq 2$ を与えたとき，$\boldsymbol{y} = (y_1,\ldots,y_\ell)^T$ に対し尤度関数

$$\frac{1}{(2\pi)^{\ell/2}(\sigma^2)^{\ell/2}|\mathbf{R}|^{1/2}} \times \exp\left[-\frac{(\boldsymbol{y}-\mu\mathbf{1})^T\mathbf{R}^{-1}(\boldsymbol{y}-\mu\mathbf{1})}{2\sigma^2}\right]$$

を最大化することによって μ および σ の推定を以下のように得る．

$$\hat{\mu} = \frac{\mathbf{1}^T\mathbf{R}^{-1}\boldsymbol{y}}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}$$
$$\hat{\sigma}^2 = \frac{1}{\ell}(\boldsymbol{y}-\hat{\mu}\mathbf{1})^T\mathbf{R}^{-1}(\boldsymbol{y}-\hat{\mu}\mathbf{1})$$

ここで，$\mathbf{1} = (1,\ldots,1)^T$ であり，$\mathbf{R}$ はその (i,j) 成分が $\mathrm{Corr}(\varepsilon(\boldsymbol{x}_i),\varepsilon(\boldsymbol{x}_j))$ となる $\ell\times\ell$ 行列である．

このとき，学習サンプル $(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_\ell,y_\ell)$ にもとづく $y(\boldsymbol{x})$ の最良線形不偏推定として次を得る [12].

$$\hat{y}(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{r}_{\boldsymbol{x}}^T\mathbf{R}^{-1}(\boldsymbol{y}-\hat{\mu}\mathbf{1}) \tag{7}$$

ただし，

$$\boldsymbol{r}_{\boldsymbol{x}} = [\mathrm{Corr}(\boldsymbol{x}_1,\boldsymbol{x}),\ldots,\mathrm{Corr}(\boldsymbol{x}_\ell,\boldsymbol{x})]^T$$

である．また，近似モデルの平均2乗誤差 $\hat{s}^2(\boldsymbol{x})$ は次式で与えられる．

$$\hat{s}^2(\boldsymbol{x}) = \hat{\sigma}^2\left(1 - \mathbf{r}_{\boldsymbol{x}}^T\mathbf{R}^{-1}\mathbf{r}_{\boldsymbol{x}} + \frac{(1-\mathbf{1}^T\mathbf{R}^{-1}\mathbf{r}_{\boldsymbol{x}})^2}{\mathbf{1}^T\mathbf{R}^{-1}\mathbf{1}}\right) \tag{8}$$

3. 追加学習点の選定

できる限り少ない関数評価回数で，十分な精度をもつ近似最適解を得るためには，少なめの初期学習サンプルから出発し，少しずつ学習サンプルを追加して近似モデルの修正を行う．逐次近似最適化においては，いかにうまく追加学習点を定めるかが非常に重要となる．

[2] これは ordinary Kriging と呼ばれ，μ は定数であるが未知としている．他に $\mu(\boldsymbol{x}) = \sum w_i h_i(\boldsymbol{x})$ とする universal Kriging 等もある

3.1 D-最適性

工学設計でよく用いられる応答曲面法では実験計画法にもとづき，線形回帰モデルにおける $\hat{\boldsymbol{w}}$ の分散をできる限り小さくするサンプル点をとる．たとえば，式 (2) で表される $\hat{\boldsymbol{w}}$ の共分散行列における $(H^TH)^{-1}$ を最小にする一つの方法として，$\det(H^TH)$ を最大にする D-最適性基準がある．この他にも実験計画法では様々な基準が提案されている [7, 11] が，これらはモデル依存性が高く，最初に設定したモデルの責任が非常に大きくなる．一般には，関数値の変化が少ない所は学習点は少なく，変化の大きい所は密に取るのがよいと考えられる．複雑な応答をもつ問題や，事前にどのようなモデルが適切か全く分からないような場合には応答曲面法で通常よく用いられている多項式モデルでは不十分である．このようなときには基底として RBF を用いればよいが，そのとき D-最適性による追加学習点の選定はなるべくデータ空間に均一に分布するように行われる [11]．これは，次に述べる空間充満法と同じになる．

3.2 空間充満法

事前に応答がどのような複雑性を持っているかわからないときには，直感的にはこれまでの学習サンプルが余りない所に追加学習点を取るのがよいと考えられる．学習点がデータ空間に偏りなく分布していることを目指すのが空間充満 (space filling) の考えである．ラテン超方格（LHC) がこの部類に属するが，LHC は追加学習点の選定には向かない．k-最近傍法を利用して既存学習サンプルの疎なところに追加学習点を選定する方法も考えられている [8].

3.3 期待改善量による方法

最適化におけるメタモデリングに対しては最適解の精度良い近似と目的関数の全体の形状の把握を目指すことになる．Jones らは EGO (efficient global optimization) と呼ばれるブラックボックス目的関数をもつ大域的最適化法において，クリギングにより導かれる期待改善量 (expected improvement: EI) によって追加学習点を定める方法を提案した [4]．EI とは，目的関数の最小値をとる可能性のある部分，または近似モデルにおける不確定性の高い（言い換えると設計空間におけるサンプル点の密度が低い）部分に高い値を与える一つの指標である．

与えられたサンプルデータ $(\boldsymbol{x}_1, y_1),\ldots,(\boldsymbol{x}_\ell, y_\ell)$ に対する現在の最良値 $f_{\min} = \min\{y_1,\ldots,y_\ell\}$ に対し，ある点 $\boldsymbol{x}$ における目的関数値 y を確率変数 Y の実現値と考えるとき，点 $\boldsymbol{x}$ における関数値が $f_{\min}$ からど

れだけ改良されたかを示す改善量は

$$I(\boldsymbol{x}) = \max\left\{0,\ f_{\min} - Y\right\} \tag{9}$$

によって与えられる．$I(\boldsymbol{x})$ の期待値（期待改善量）は式 (7) で与えられる $\hat{y}$，および式 (8) で与えられる $\hat{s}$ に対し，次式によって計算される：確率変数 Y は $N(\hat{y}, \hat{s}^2)$ に従うものとし，ϕ をガウス密度関数，Φ をその分布関数とするとき，$\hat{s} \neq 0$ に対し

$$\begin{aligned} &\mathrm{E}\big[I(\boldsymbol{x})\big] \\ &= \mathrm{E}\left[\max\left\{0,\ f_{\min} - Y\right\}\right] \\ &= (f_{\min} - \hat{y})\,\Phi\left(\frac{f_{\min} - \hat{y}}{\hat{s}}\right) + \hat{s}\phi\left(\frac{f_{\min} - \hat{y}}{\hat{s}}\right) \end{aligned} \tag{10}$$

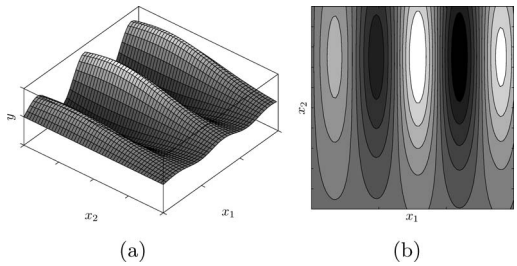
また，$\hat{s} = 0$ の場合は $\mathrm{E}[I] = 0$ とする．EI の第 1 項は最適解の可能性の大小を測り，第 2 項は不確定性の大小を測っている．したがって，最適解の可能性の高い所は第 1 項が，また設計空間におけるサンプル点が疎なところは第 2 項が利くことによって選ばれる．

EGO では EI が最大になるような点を追加サンプル点として採用する．EI を用いる方法は逐次近似最適化において魅力的であるが，その計算に少し負荷がかかるという難点がある．

3.4 局所情報と大局情報による追加法

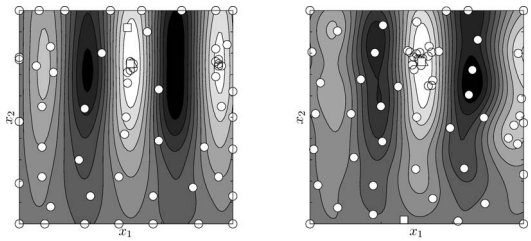
最適化におけるメタモデリングで重要なことは最適解が存在すると思われる所を精査し，同時に目的関数全体の形状を把握することである．そこで，この目的のために，より精度の高い近似最適解を求めるための局所的情報と，精度良い近似目的関数を得るための大局的情報を同時に追加していく方法が提案されている [8]．一つ目は，現在の最適点付近の詳細な情報を得るために現在の最適解付近の点を，二つ目は局所解への収束を防ぐためにこれまでのサンプル点の分布が疎な部分の点を追加する．この方法は局所情報と大局情報を同時に追加していくことによって逐次近似最適化の効率を上げている．また，設計空間におけるサンプル点間の距離計算だけで済むので計算が簡単であり，さらにパラメータチューニングが不要であることなどの特徴がある．

図 1 はある問題の正確な目的関数の形状と等高線を示している．ここで，問題は目的関数の最大化で最適解は中央の山脈の頂点である．図 2 はこの問題に対し，初期 5 点から追加学習していって EI による追加で学習点が 63 点のときの結果と大局情報と局所情報を同時に追加する方法による追加で学習点が 61 点のときの結果を示している．どちらも全体をみながらも解になる可能性のある部分を詳細に追加していっている様子がわかる．



(a) (b)

図 1 正確な目的関数の形状 (a) と等高線 (b)



(a) (b)

図 2 EI による結果 (a) と大局情報と局所情報を同時に追加する方法による結果 (b) の比較

4. 逐次近似多目的最適化手法

工学設計などの実問題ではあれもこれもよくしたいという多目的最適化問題

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_r(\boldsymbol{x}))^T \\ \text{s.t.} \quad & \boldsymbol{x} \in X \subset \mathbb{R}^n \end{aligned}$$

になることが多い．多目的最適化問題ではすべての目的関数を同時に最適化する解が必ずしも存在せず，各目的関数の効率をぎりぎりまで高めた Pareto 解の概念が導入される．通常，Pareto 解は多数存在し，その中から意思決定者の価値判断にあう解（意思決定解）を選ぶことになる．多目的最適化におけるこれまでの研究には

(i) 意思決定者の価値判断に関する情報を引き出しながらできる限り早く，また容易に意思決定解を見いだそうとする対話型解法

(ii) Pareto フロンティア（目的関数空間における Pareto 解集合）を提示して，後は意思決定者の判断に任せる

という二つのアプローチがある．(i) のアプローチで代

表的なものに，意思決定者の価値基準の情報として希求水準が与えられたときに，それに最も近い Pareto 解を提示し，その解に満足できないときは希求水準を変更して同様の手続きを繰り返すという希求水準法がある [10]．(ii) のアプローチは遺伝的アルゴリズムなどのメタヒューリスティク手法や DEA を用いて，近年盛んに研究が行われている [1, 3, 9]．近似 Pareto フロンティア生成においては，真の Pareto フロンティアになるべく早く近づかせること（収束性）および Pareto 解をなるべく広く分布させること（解の多様性）が重要な課題となり，そのために様々な工夫やメカニズムが研究されている．

多目的意思決定において，目的関数が二つの場合には，Pareto フロンティアを図示すれば，目的間のトレードオフは一目瞭然で意思決定者は容易に意思決定解を定めることができる．しかし，目的関数が三つになると，Pareto フロンティアを図示することはできても，意思決定者が目的関数のトレードオフを読み取ることはそれほど容易ではなくなり，四つ以上になるともはや視覚化することはできなくなる．

とはいえ，意思決定者に一つの解を提示するだけでなく，その近傍の情報やおおざっぱでも全体の概観を示すことができれば意思決定に非常に役立つ．そこで，意思決定者にとって最も関心が深いと思われる Pareto 解もしくは Pareto フロンティア上でのその近傍を精度良く近似し，Pareto フロンティアの全体像はそこそこの精度で近似するという方法が現実的と考えられる．

意思決定者にとって最も関心の深いのは自分の価値基準に見合った解であるので，希求水準法によってそれを効率よく見つけ，同時にその近傍の Pareto 解の状況さらには Pareto フロンティア全体の概況を把握するという上記 (i)，(ii) を融合した手法も考えられている [14]．以下にこの方法を簡単に紹介しよう．

Step 1. 学習サンプル $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_\ell, y_\ell)$ にもとづいて近似目的関数 $\hat{f}_k(\boldsymbol{x})$，$k = 1, \ldots, r$ を適当なメタモデリング手法によって求める．

Step 2. $\hat{f}_k(\boldsymbol{x})$，$k = 1, \ldots, r$ に対し，次の拡大 Tchebyshev スカラー化関数

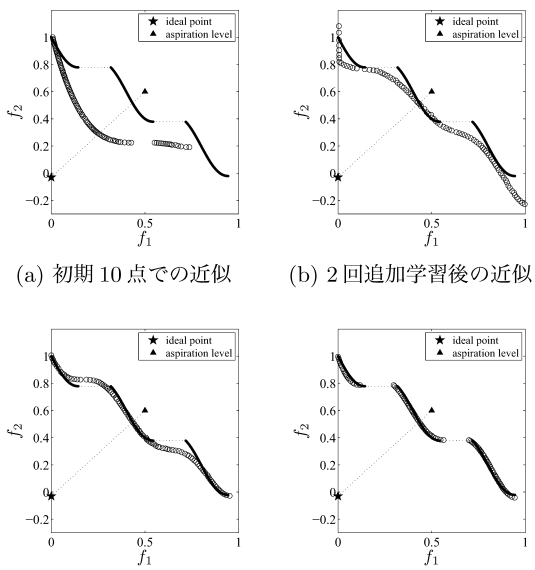
$$F(\boldsymbol{x}) := \max_{1 \leqq k \leqq r} w_k \big(\hat{f}_k(\boldsymbol{x}) - \overline{f}_k\big) + \alpha \sum_{i=k}^{r} w_k \hat{f}_k(\boldsymbol{x})$$

を最小化することによって希求水準 $\overline{f}_k$，$k = 1, \ldots, r$ に対し最も近い Pareto 解を求める．ここで，f_k^* を理想点とするとき，重みは $w_k = 1/(\overline{f}_k - f_k^*)$ によって与える．

同時に $\hat{f}_k(\boldsymbol{x})$，$k = 1, \ldots, r$ に対し，適当な方法によって近似 Pareto フロンティアを生成する．

Step 3. 必要あれば学習サンプルを追加して近似モデルの精度を上げる．このとき，追加学習点の選定は単目的の時のように単純ではなくなり，解の概念が広がるため種々の追加点選定の方法が提案されている．意思決定者にとって最も興味のある解もしくはその近傍として，Step 2 で得られた解もしくはその近傍の点を一つの追加点とし，さらに近似モデルの精度を上げるために既存の学習サンプルの疎なところに二つ目の追加点をとる．これだけでもよいが，さらに Pareto フロンティア全体の近似精度向上のためにランク関数を近似しながらその最大点を与える方法 [11] や Pareto 適応度関数を近似しながらその最大点を追加する方法 [5] 等も併せ用いるとより効率が上がる．

図 3（図中の ∘ は近似 Pareto 解，実線は真の Pareto 解を表す）はある問題に対し，ランダムに与えた初期 10 点での近似の様子と，3 点ずつ追加していったときの近似の様子を表している．理想点と希求水準を結ぶ直線と近似 Pareto フロンティアとの交点が上記 Step 2 で得られる Pareto 解である．2 回追加学習後（学習点：16）では近似目的関数に対し Step 2 で得られる Pareto 解は真の目的関数に対する Pareto 解を十分な精度で近似し，その解の近傍で Pareto フロンティアも一定の精度で近似できている．6 回追加学習後（学



(a) 初期 10 点での近似　(b) 2 回追加学習後の近似

(c) 6 回追加学習後の近似　(d) 10 回追加学習後の近似

図 3　希求水準法の解と近似 Pareto フロンティア

習点：28）では相当広い範囲で Pareto フロンティアが精度良く近似でき，10 回追加学習後（学習点：40）では Pareto フロンティア全体が十分な精度で近似できている．類似の方法として ParEGO [6] が提案されているが Pareto フロンティア全体の近似を目的としたもので，意思決定者の価値判断は取り入れていない．

5. おわりに

本解説では，ブラックボックスである目的関数の形を予測しながら同時に最適化を行っていく逐次近似最適化法に計算知能の技法が効果的に応用できることを紹介した．実際の問題，とくに工学設計問題ではシミュレーションや実験によって始めて関数の値がわかることが多く，なるべく少ない実験回数で解を得るというこのようなアプローチは実用的な方法としてニーズが高まってきている．多くの応用とともに手法がますます洗練されていくことが期待される．

参考文献

[1] C. A. C. Coello, D. A. Van Veldhuizen and G. B. Lamont: *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers (2001).

[2] N. Cristianini and J. Shawe-Taylor: サポートベクターマシン入門, 大北　剛（訳），共立出版 (2005).

[3] K. Deb: *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Co. Ltd. (2001).

[4] D. R. Jones, M. Schonlau and W. J. Welch: Efficient global optimization of expensive black-box functions, *Journal of Global Optimization*, Vol. 13, pp. 455–492 (1998).

[5] 北山哲士，荒川雅生，山崎光悦: RBF ネットワークによる多目的逐次近似最適化，日本機械学会論文集 C 編，76–772，pp. 3476–3485 (2010).

[6] J. Knowles: ParEGO: A hybrid algorithm with online landscape approximation for expensive multiobjective optimization problems, *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 1, pp. 50–66 (2006).

[7] R. Myers and D. Montogomery: *Response Surface Methodology*, Wiley (1995).

[8] H. Nakayama, M. Arakawa and R. Sasaki: Simulation-based optimization using computational intelligence, *Optimization and Engineering*, Vol. 3, No. 2, pp. 201–214 (2002).

[9] 中山弘隆，岡部達哉，荒川雅生，尹　禮分: 『多目的最適化と工学設計—しなやかシステム工学アプローチ—』，現代図書 (2007).

[10] 中山弘隆: 多目的計画法に対する満足化トレードオフ法の提案，計測自動制御学会論文集，Vol. 20, pp. 29–35 (1984).

[11] H. Nakayama, Y. Yun and M. Yoon: *Sequential Approximate Multiobjective Optimization Using Computational Intelligence*, Springer (2009).

[12] J. Sacks, W. J. Welch, T. J. Mitchell and H. P. Wynn: Design and Analysis of Computer Experiments, *Statistical Science*, Vol. 4, No. 4, pp. 409–435 (1989).

[13] B. Schölkopf and A. J. Smola: *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*—, MIT Press (2002).

[14] 尹　禮分，中山弘隆，尹　敏: 計算知能を用いた逐次近似多目的最適化手法，計測自動制御学会論文集，Vol. 43, No. 8, pp. 672–678 (2007).