Vol.2017-HPC-159 No.2
2017/4/17

遺伝的アルゴリズムに基づく疎行列解法のパラメタに関するオンライン自動チューニング

山本 堅太郎[1]　松本 正晴[1]　須田 礼仁[1]

概要：遺伝的アルゴリズム（GA）によって GMRES の初期値を生成して性能を改善する手法が直野・佐川らによって提案されている．本論文では同様の GA を BiCG に適用し，GA に関するオプションをオンライン自動チューニングで探索する手法を提案する．提案手法は須田らによるワンステップ近似に基づいているが，反復法は収束しないことがあり，コストが無限大になるという問題があった．それに対して本研究では反復法が収束する確率をモデル化してワンステップ近似を拡張した．またラテン格子を用いた初期実験で効率的に性能を推定した．その結果，提案手法は単純な自動チューニングに比べ最大で約 2 倍の性能を達成した．

Online Autotuning of Parameters of Genetic Algorithm-based Sparse Linear Solver

YAMAMOTO Kentaro[1]　MATSUMOTO Masaharu[1]　SUDA Reiji[1]

1. はじめに

連立一次方程式の求解，特に偏微分方程式を解く際に頻繁に表れる大規模疎行列を係数とする連立一次方程式の求解は科学技術計算の根本要素であり，高速・高並列・高安定なアルゴリズムの探求が行われ続けている．近年は前処理付き Krylov 部分空間法やマルチグリッド法など収束性のよい解法がよく研究されている．

その中で直野ら [1], 佐川 [2] による GA-GMRES は，遺伝的アルゴリズム (GA) を Krylov 部分空間法の 1 つである GMRES に適用した特徴的な研究である．GA-GMRES では，GMRES の初期ベクトルを GA で決める．複数の初期ベクトルから比較的少ない回数の GMRES 反復を行い，（比較的精度の低い）近似解を複数得る．これに交差や突然変異などの GA 的操作を施し，次の複数の初期ベクトルを作成する．これを繰り返すと，急速に解に収束する初期値が得られるというものである．

我々は GA-GMRES に興味を持ち，同様の手法を BiCG に適用してみた．これを GA-BiCG と呼ぶことにする．しかし思うような結果はなかなか得られなかった．GA-BiCG にはアルゴリズムの挙動に影響を与えるいくつものパラメタがある．例えば，GA における個体数，上述の GA-GMRES で説明した「比較的少ない回数の GMRES 反復」の反復数（以下，**1 世代あたりの反復回数**という），交差や突然変異などの GA 的操作などである．従来研究でもこれらのパラメタが収束に与える影響が議論されているが，我々の予備実験でもこれらのパラメタの影響は明確であり，また行列によっても影響が異なる様子であった．そして，行列ごとに GA-BiCG のための最適パラメタを人手で決定するのは容易ではない．

そこで，本研究では，自動チューニングの手法を GA-BiCG に適用し，行列ごとにチューニングパラメタを自動的に最適化することを目指した．ここでは，チューニングパラメタとして，GA における個体数 nc，1 世代あたりの反復回数 ITE, 交差の方法 co を選んだ．また，我々が以前から取り組んでいる Bayes 統計に基づく近似最適逐次実験計画法であるワンステップ近似 [3] をベースにして，反

1　東京大学情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo, 7–3–1 Hongo, Bunkyo, Tokyo 113–8656, Japan

復回数を目的関数とするオンライン自動チューニングの手法を開発した．

本研究で自動チューニングとしては新たな課題が1つ生じた．ワンステップ近似は一般の分布に単純に拡張できるが，正規分布は特によい性質を示すことが知られている [4]．今回は BiCG の反復回数をコストとしているが，一般に反復法は行列や初期値によっては（設定された上限反復回数までに）収束しない場合もある．収束しない場合に反復数を「上限反復回数」とすると，実際よりも下に見積もりすぎになるのに加えて，上限反復回数に値が集中してしまい，正規分布から大きくはずれてしまう．残差が増加してゆくような場合には反復回数は「無限大」とするのが適切と思われるが，そのように扱うと平均も分散も発散して正規分布は完全に使えなくなってしまう．

そこで我々はコストである反復回数だけではなく，「収束するかしないか」を表すもう1つのコスト関数を定義し，これらの組み合わせで最適化することとした．行列と GA のパラメタを固定しても，右辺が変わると，収束したりしなかったりする．我々は右辺は確率的に定まると想定して，「収束する場合の反復回数」と「収束確率」を別々に推定することにした．提案手法では，反復回数が上限に達した時には，そこまでの結果を破棄して，パラメタを最も収束しやすいと推定されるものに変更して求解を最初からやり直す．それでも収束しない場合には，どのようなパラメタでも収束しない恐れがあると考えて，求解そのものを諦めることとした．すなわち最悪の場合には上限反復回数の2倍の反復が行われることになる．このような整理のもとで反復回数の Bayes モデルを構築することができる．さらにワンステップ近似を適切に拡張することで，オンライン自動チューニングを実現することができた．

本研究ではさらなる高性能化として，パラメタごとに性能に与える影響を推定するモデルを構築し，これを利用する．このモデルはワンステップ近似によるオンライン自動チューニングに入る前に，ラテン格子を用いた初期実験を行うことで初期化される．これにより少数の初期実験で性能の大体の傾向をつかむことができ，自動チューニングを効率化することができた．

以下，本論文は次のような構成となっている．次の2節では自動チューニングの対象である GA-BiCG を説明する．3節では自動チューニング手法を説明し，4節で結果について述べる．5節は本論文のまとめである．

2. GA-BiCG

図1に GA-BiCG のアルゴリズムを示す．各世代での反復回数の上限 ITE, 交差アルゴリズム co, 個体数 nc は与えられるものとする．返り値は計算に要した反復回数である．全世代を通算した総反復回数の上限 β も行列サイズを参照して設定されているものとする．また本論文では収束条件として相対残差が 10^{-8} となったところで反復を打ち切るものとした．

ベクトル $x[k]_i^l$ は第 l 世代の k 番目の染色体による BiCG の 第 i 反復における近似解，ベクトル $x^*[k]_i^l$ は対応するシャドーベクトルである．まず，右辺 b^* および $\left(x[k]_0^1, x^*[k]_0^1\right)$ は乱数で生成する．次に l ループが GA の世代に関する反復である．そこではまず，nc 個ある染色体をそれぞれ初期値として BiCG を ITE 回まで反復する．そうして得られた近似解の中で残差が 10^{-8} より小さいものが1つでもあれば，解が得られたとして終了する．また $l \times ITE$ が β より大きくなれば，収束しなかったものとしてやはり計算を終了する．それ以外の場合は，最後の近似解の集合に交差を施して新たな初期値を生成する．

交差のアルゴリズムは4つある．これらは佐川 [2] と同じものを採用した．

0 Simple average : 2つのベクトルの和を2で割る．

1 Single point crossover : 新たなベクトルの最初の半分の要素は1つのベクトルから，残りの半分の要素はもう1つのベクトルから取る．

2 Mutated simple average : Simple average の結果に小さな乱数を加える．乱数の大きさは残差ノルムの 1/10 とする．

3 Weighted average : 新たなベクトルの残差が最小となるように，2つのベクトルの重み付き平均を取る．

ただし weighted average は具体的には次のようにした．反復最後の $\left(x[k]_{ITE}^l, x[k+1]_{ITE}^l\right)$ から

$$
\begin{aligned}
x[k]_0^{l+1} &= x[k]_{ITE}^l + t\Delta x, \\
x^*[k]_0^{l+1} &= x[k]_{ITE}^l + t\Delta x^*, \\
\Delta x &= x[k+1]_{ITE}^l - x[k]_{ITE}^l, \\
\Delta x^* &= x^*[k+1]_{ITE}^l - x[k]_{ITE}^l, \\
t &= \mathrm{argmin}\|b - Ax[k]_0^{l+1}\|_2 \\
&= \left(b - Ax[k]_{ITE}^l, A\cdot\Delta x\right) / \left(A\cdot\Delta x, A\cdot\Delta x\right).
\end{aligned}
$$

のようにして $\left(x[k]_0^{l+1}, x^*[k]_0^{l+1}\right)$ を計算する．

3. GA-BiCG の自動チューニング手法の提案

3.1 自動チューニングの概要

図2に GA-BiCG の自動チューニングの概要を示す．これはオンライン自動チューニングの問題設定であり，係数行列が同じ A で右辺 b が異なる N 個の連立一次方程式を逐次に解く．各求解（以下**ステップ**という）で下記のオプションを設定しながら，N ステップの合計の反復回数をできるだけ小さくしたい．

各回の計算で**オプション**と呼ぶ (ITE, co, nc) の組を選ぶ．本研究では，反復回数 ITE は 50, 100, 150, 200 のうちから1つ，交差 co のアルゴリズムは 0, 1, 2, 3 のうちから1つ，個体数 nc は 1, 4, 8, 12 のうちから1つを選ぶこ

```
Set initial guess pairs of vectors (x[k]_0^1, x*[k]_0^1) with random numbers
Set the constant vector b* with random numbers
For l = 1, 2, ... , Do
    For k = 1, 2, ...nc, Do
     Solve Ax = b with initial guess (x[k]_0^l, x*[k]_0^l) using BiCG until ITE steps
    EndDo
    If any relative residual norm is smaller than 10^-8
        Return l × ITE
    EndIf
    If l × ITE is bigger than β
        Return β
    EndIf
    Cross (x[k]_ITE^l, x*[k]_ITE^l) (k = 1, 2, ..., nc) to obtain (x[k]_0^{l+1}, x*[k]_0^{l+1}) with crossover algorithm co
EndDo
```

図 1 GA_BiCG(ITE, co, nc) のアルゴリズム

```
For i = 1, ..., N Do
    Choose option o = (ITE, co, nc)
    Solve Ax_i = b_i using GA_BiCG(ITE, co, nc)
    If the algorithm fails
        Choose another option o' = (ITE', co', nc')
        Solve Ax_i = b_i using GA_BiCG(ITE', co', nc')
    EndIf
    Update option o and o'
EndDo
```

図 2 GA-BiCG の自動チューニングの概要

とにした．これらの組み合わせにより，合計 64 通りのオプションがある．

3.2 収束確率の推定

ワンステップ近似では，正規分布を仮定した Bayes 統計を用いてきた．ところが第 1 節で詳述したように，反復が収束しない場合があるため，正規分布の仮定が明らかに適切でない．

そこで本研究では，反復回数のモデルに加えて，収束する「確率」をモデル化することとした．すなわち，反復が収束する場合としない場合が確率的に発生すると仮定して「収束する確率」（以下「収束確率」という）を推定するモデルを作る．また反復が収束する場合に限り，その反復回数を推定するモデルを作る．これら 2 つのモデルを自動チューニングの進行に従って更新しつつ，よいオプションを探る．

反復回数は従来研究 [3] と同様に正規分布で推定する．事前分布については後述する．以下では収束確率のモデルについて説明する．

オプション A の収束確率を θ_A とする．本研究では θ_A はベータ分布に従うものと仮定し，事前分布を $Be(a,b)$ とする．ここで a, b は非負実数である．自動チューニングの実行の中でオプション A が n 回選ばれて，そのうち k 回が収束したとする．このとき θ_A の事後分布の確率密度関数は

$$p(\theta_A) = \frac{\theta^{k+a-1}(1-\theta)^{n-k+b-1}}{B(k+a, n-k+b)}$$

となる．これは $Be(k+a, n-k+b)$ というベータ分布であり，収束確率の期待値は $p = \frac{k+a}{n+a+b}$ である．

3.3 ワンステップ近似の拡張

次に，収束確率の推定モデルを利用してワンステップ近似を拡張する．

我々は，オプション A のコスト c_A を以下のように定義する．

$$c_A = \theta_A \mu_A + (1 - \theta_A)\, 2\beta$$

ここで θ_A は収束確率のモデル，μ_A はモデルにより推定された反復回数の期待値である．コスト c_A が意味するところは，確率 θ_A で収束し，収束したら反復回数 μ_A がかかり，収束しなければ反復回数 2β がかかるということである．収束しなかった場合は，1 回は β の反復がかかるが，もう 1 回代替のオプションを用いて再度解こうとするため，それが収束すれば β より少ない反復回数となる．しかし，我々はそもそもオプション A で 1 回解けなかった問題はオプションを変えても解けない可能性が高いと考えた．そのため，最悪値である 2β でモデル化することはある程度妥当であると考えた．

ここで，まず第 $N-1$ ステップ（最後から 2 ステップ目）の実行におけるオプションの選択について議論する．第 $N-1$ ステップにオプション A が実行されて，収束した場合には反復回数が x_A となると仮定する．この x_A は第 $N-1$ ステップの実行の前の事前分布により推定される．その確率密度関数を $q_A(x_A)$ とする．

このとき，最後の第 N ステップの反復にかかるコストの期待値は以下のように推定される．

$$
\begin{aligned}
\hat{x}_{AN} &= \int_0^1 (1-\theta)\, c' p(\theta)\, d\theta + \int_0^1 \theta c'' p(\theta)\, d\theta \\
c' &= \begin{cases} \bar{c}_A & \text{if } \bar{c}_A < \bar{c}_{min} \\ \bar{c}_{min} & \text{otherwise} \end{cases} \\
c'' &= \int_{-\infty}^{\xi_A} c_A q_A(x_A)\, dx_A \\
&\quad + \int_{\xi_A}^{\infty} \bar{c}_{min} q_A(x_A)\, dx_A \\
\bar{c}_A &= \int_0^1 c_A p(\theta)\, d\theta \\
&= \frac{k+a}{n+a+b+1}\mu_A + \left(1 - \frac{k+a}{n+a+b+1}\right) 2\beta \\
\bar{c}_{min} &= \min_{i \neq A} c_i
\end{aligned}
$$

右辺第1項は第 $N-1$ ステップにオプション A で反復が収束しなかった場合に対応する．このとき，収束確率のモデルは更新されるが，反復回数のモデルは更新されない．この更新の結果オプション A のコストの期待値が $\bar{c}_A$ となる．最終回には，コストが最小のオプションを選ぶべきであるが，そのコストが c' である．

右辺第2項は第 $N-1$ ステップにオプション A で反復が収束した場合に対応する．このとき，反復回数のモデルは更新されるが，収束確率のモデルは更新されない．この更新の結果は反復回数 x_A による．式中に現れる ξ_A は，$x_A \leq \xi_A$ であれば，オプション A のコストの期待値が，全オプションの中で最小になるような値である．すなわち $x_A \leq \xi_A$ であれば最終回にはオプション A が選ばれ，そうでなければ A 以外のオプションでコストの期待値が最小となるオプションが選ばれる．その結果，最終回のコストの期待値が c'' となる．

我々は $p(\theta)$ の扱いについていくつかの方法を試した．その結果，もっとも単純でありながらもっとも有効だった手法は，$\int \theta p(\theta) d\theta$ のかわりに θ の期待値 p_A を用いるものであった．このときコストの期待値は

$$c_A = p_A \mu_A + (1-p_A) 2\beta$$

となる．オプション A が n 回実行されて k 回収束したとき，$p_A = \frac{k+a}{n+a+b}$ となる．このとき最終回のコストの期待値 $\hat{x}_{AN}$ は

$$
\begin{aligned}
\hat{x}_{AN} &= E_1 + E_2 \\
E_1 &= (1-p) \min\{(p'\mu_A + (1-p')\, 2\beta), \bar{c}_{min}\} \\
E_2 &= p \int_{-\infty}^{\xi_A} (p''\mu_A + (1-p'')\, 2\beta)\, q(x_A)\, dx_A \\
&\quad + p \int_{\xi_A}^{\infty} \bar{c}_{min} q_A(x_A)\, dx_A
\end{aligned}
$$

のようになる．ここで確率 p' と p'' はそれぞれ $\frac{k+a}{n+a+b+1}$ と $\frac{k+a+1}{n+a+b+1}$ となる．これらをまとめると

$$
\begin{aligned}
E_1 &= \left(1 - \frac{k+a}{n+a+b}\right) \min\{D, c_{min}\} \\
E_2 &= A + 2\beta B + C \\
A &= \frac{(k+a)(k+a+1)}{(n+a+b)(n+a+b+1)} \int_{-\infty}^{\xi_A} \mu_A(x_A)\, q_A dx_A \\
B &= \frac{(k+a)(n-k+b)}{(n+a+b)(n+a+b+1)} \int_{-\infty}^{\xi_A} q_A(x_A) dx_A \\
C &= \frac{k+a}{n+a+b} \bar{c}_{min} \int_{\xi_A}^{\infty} q_A(x_A) dx_A \\
D &= \frac{k+a}{n+a+b+1}\mu_A + \left(1 - \frac{k+a}{n+a+b+1}\right) 2\beta
\end{aligned}
$$

となる．

```
For i = 1, ..., N , Do
    Choose option o_i = (ITE_i, co_i, nc_i)
    Solve Ax_i = b_i using GA_BiCG(ITE_i, co_i, nc_i)
    If the algorithm fails
        Choose another option o_i' s.t. o_i' = argmax_{j≠i} p_j
        Solve Ax_i = b_i using GA_BiCG(ITE_i', co_i', nc_i')
        If the algorithm fails
            update p_i
            update p_i'
        Else
            update p_i
            update p_i', μ_i', τ_i'^2
        EndIf
    Else
        update p_i, μ_i, τ_i^2
    EndIf
EndDo
```

図 3 自動チューニングにおける推定値の更新

3.4 収束確率の更新

自動チューニングにおいてどのような情報更新が行われるかを図3に示す．提案手法では，選んだオプションで反復が収束しなかった場合，収束確率が最も高いと推定されるオプションで解きなおす．

本研究では，解きなおしの際に収束したか否かの情報を，解きなおしに選ばれたオプションの収束確率に反映させている．最初のオプションで収束しなかったということは，そもそも解きにくい問題である可能性があるので，解きなおしのオプションの収束確率に反映させるのが適切かどうかは議論の余地がある．しかし本研究では，より詳細な情報を収集することに重きを置いて，情報を更新することとした．

3.5 初期実験

何ら事前情報がない場合，自動チューニングにできることはランダムにオプションを選ぶことぐらいになってしまう．本研究では，少数の初期実験によりオプションの効果

を推定することとした．具体的には，全部で N ステップ解かれる連立一次方程式の最初の数ステップを決まったオプションで解く「初期実験」に当て，残りのステップでワンステップ近似を用いたオンライン自動チューニングを実行する．すなわち，全体の実行ステップ数は変わらず，計算の最初の部分が初期実験にあてられる．

本研究では初期実験のための実験計画にラテン格子の一種である Symmetric Latin Hypercube Design (SLHD) [5] を用いた．これを GA-BiCG に用いると，例えば以下のような標本点が得られる．

$$\begin{bmatrix} (50,2,8) \\ (50,0,1) \\ (100,0,4) \\ (100,1,12) \\ (150,2,1) \\ (150,3,8) \\ (200,3,12) \\ (200,1,4) \end{bmatrix}$$

3.6 初期実験を用いた性能の推定

前節で導入された初期実験 $o_i = (ITE_i, co_i, nc_i)$ $(i = 1,..,8)$ から性能を以下のように推定する．以下ではオプション o_i による反復回数を x_i とし，またオプション o_i による反復が収束したとき $s_i = 1$, 収束しなかったとき $s_i = -1$ とする．

まず，実験的に，オプションのうち co が性能に与える影響は ITE と nc よりも小さいことを観測した．そこで，我々は ITE と nc のみを性能推定に用いることとした．モデルには線形モデル

$$\begin{aligned} x &= w_1 \cdot ITE + w_2 \cdot nc + w_3 \\ s &= u_1 \cdot ITE + u_2 \cdot nc + u_3 \end{aligned}$$

を用いた．係数 (w_1, w_2, w_3) および (u_1, u_2, u_3) は線形最小二乗法により推定した．ここから事前分布のパラメタ $\mu_{i,0}$ と $p_{i,0}$ を

$$\begin{aligned} \mu_{i,0} &= w_1 \cdot ITE_i + w_2 \cdot nc_i + w_3 \\ p_{i,0} &= \begin{cases} 1 & \text{if } u_1 \cdot ITE_i + u_2 \cdot nc_i + u_3 > 0 \\ \frac{1}{2} & \text{otherwise} \end{cases} \end{aligned}$$

のように設定した．ここで $p_{i,0}$ は $\frac{1}{2}$ または 1 の値を取る．これに 2 回の事前実験に相当する重みを与えた．すなわち，2 回の事前実験のうち 1 回は収束したものとし，もう 1 回は $s_j > 0$ なら収束で，そうでなければ収束しなかったものとみなす．

また，τ^2 と σ^2 は以下のようにして定めた．初期実験における反復回数の標本分散が

Set $n_i = 0$
Set $M_i = 0$
Set $S_i = 0$
For each x_i^j observed at j th execution with option in O_i, Do
  If $\frac{x_i^j - M_i}{\sqrt{\sigma^2_{i,j-1}}} < \gamma$
    $n_i = n_i + 1$
    $M_{old} = M_i$
    $M_i = M_i + (x_i^j - M_i)/n_i$
    $S_i = S_i + (x_i^j - M_i)(x - M_{old})$
    $\sigma^2_{i,j} = S_i/(n_i - 1) + (ITE_i)^2/12$
  EndIf
EndDo

図 4 $\sigma^2_{i,j}$ の推定方法

$$\begin{aligned} \sigma_{ini} &= \frac{1}{|S|-1} \sum_{j \in S} (x_j - \bar{x})^2 \\ \bar{x} &= \frac{1}{|S|} \sum_{j \in S} x_j \\ S &= \{ j \mid \text{オプション } o_j \text{ が収束} \} \end{aligned}$$

と計算される．この σ_{ini} の期待値は $\tau^2 + \sigma^2$ となる．そこでこの値を τ^2 と σ^2 に均等に割り振ったものとして，

$$\tau^2 = \sigma^2 = \frac{1}{2} \left(\frac{1}{|S|-1} \sum_{j \in S} (x_j - \bar{x})^2 \right)$$

とした．

以上，本節では初期実験からの性能の推定方法を説明したが，これらの情報は，観測がひとつもない場合にのみ適用される．ひとつでも観測されたのちには，観測値のみを用いて性能を推定する．すなわち，オプション o_i で n 回実行して反復回数 $x_i^1, ..., x_i^n$ が得られたとして，

$$\begin{aligned} \mu_{i,n} &= \frac{1}{n_i} \sum_{j=1}^{n} x_i^j \\ \tau^2_{i,n} &= \frac{\sigma_i^2}{n} \end{aligned}$$

を事後分布に用いた．

3.7 分散の推定

分散 σ_i^2 の推定は易しくない問題である．我々は実験的に ITE と nc が共通のオプションに対しては分散が似たような値を取ることを見出した．そこで ITE と nc が共通のオプションを o_{i1}, o_{i2}, o_{i3}, o_{i4} としてこれらのオプションをからなるグループ O_i $(i = 1, .., 16)$ を

$$O_i = \{o_{i1}, o_{i2}, o_{i3}, o_{i4}\},$$

と定義する．また，$\sigma^2_{i,j}$ を j 個の観測から推定された σ_i^2 の値として，これらを Welford algorithm [6] で推定することとした．図 4 にそのアルゴリズムを示す．

反復回数 x_i^j の分布は右の裾が長く，左の裾が短い．そ

こで本研究では外れ値を検出してそれを含まないように推定することにした．この処理は図 4 の 5 行目にあり，$\gamma = 3$ とした．

また，今回の手法では反復回数が ITE の倍数に制限される．これが見かけの分散を抑える効果を持っており，観測値としては分散が 0（常に同じ反復回数だったとき）となる．これはワンステップ近似のアルゴリズムで問題を引き起こす．本研究では反復回数が ITE の倍数になってしまうことを考慮して図 4 の 10 行目にあるように，$ITE_i^2/12$ を人工的に加えてある．この加えた項は，ITE_i の幅の中で一様分布をしたときに得られる分散である．

3.8 収束確率の事前分布の更新

収束確率 p_i の事前分布の期待値 $p_{i,0}$ は，何の情報もなければ 1/2 とすることが多い．しかし何度か観測をした後には，事前分布を更新することが有効である．以下では j 個の観測後のオプション i の収束確率の事前分布を $p_{i,0,j}$ とする．そしてグループ O_i の収束確率 P_i を

$$P_i = \frac{K_i}{N_i}$$
$$K_i = k_{i1} + k_{i2} + k_{i3} + k_{i4}$$
$$N_i = n_{i1} + n_{i2} + n_{i3} + n_{i4}$$

と定義する．そして線形モデル

$$P = v_1 \cdot ITE + v_2 \cdot nc + v_3$$

を仮定して，重み付き最小二乗法で (v_1, v_2, v_3) を推定する．オプショングループ O_i に対する重みは N_i とした．

そのうえで，オプション $o_i = (ITE_i, co_i, nc_i)$ の収束確率の事前分布の期待値を

$$p_{i,0,N_i} = v_1 \cdot ITE_i + v_2 \cdot nc_i + v_3.$$

とした．これが 2 回分の観測に値するとすることにより，

$$p_{i,n_i} = \frac{2p_{i,0,N_i} + k_i}{2 + n_i}$$

がオプション i の事後分布の期待値となる．

4. 結果

実験には Intel Core i7 2630QM 2.00GHz を用いた．係数行列には Florida Sparse Matrix Collection から収束性の異なる 8 つの行列を選んだ．これらの行列を表 1 に示す．表には 100 回の実行の平均値によりもっともよいオプションともっとも悪いオプションでの収束確率 p, 平均反復回数 μ, 反復当たりのコストの平均値 c も示してある．これに見るように，行列によって最適なオプションは異なり，よいオプションと悪いオプションで性能が大きく異なることも確認できる．

実験は以下のように行った．右辺 b は各要素が 10^{-1} から 10^1 までの一様乱数で生成し，シャドウ b^* は b と同じとした．初期値 (x_0, x_0^*) も同じ分布の乱数で生成した．反復回数の上限 β は行列サイズの 1.3 倍とした．全体の実行ステップ数 N は 50, 100, 200 の 3 通りを試した．性能の指標として，次に示す ave と $stddev$ を用いる．

- ave : 総実行コストを N で割ったもの 5 つの平均値
- $stddev$: 総実行コストを N で割ったもの 5 つの標準偏差

また，以下で定義される regret と loss を計算した．

- regret: 最適なオプションの平均コストと ave との差
- loss: 最適なオプションの平均コストと，自動チューニングで最適と推定されたオプションの平均コストの差

オンライン自動チューニングの目的関数は regret の最小化である．loss が最小と regret が最小とは異なる．loss を小さくするには性能が不明確なオプションを多く試すべきであるが，そうすると実験のコストがかさみ regret としては劣化する．

またチューニング履歴として各ステップで選ばれたオプションを図示する．これらの図では縦軸はオプションの番号，横軸はステップ数を示す．各ステップでは最大 2 回の実行が行われる．最初の実行で収束した場合には赤の○，収束しなかった場合には緑の×で示す．また 2 度目の実行で収束した場合には青の○，収束しなかった場合にはピンクの×で示す．

表 2 および表 3 に実験結果を示す．表中 Pre-Ex は提案手法で測定後も初期実験で得られた事前分布を用いるもの，NIPD は提案手法で測定後は測定値のみから事後分布を決めるものである．また naive は比較のための単純な自動チューニングであり，最初に 64 のオプションを 1 回ずつ実行し，その後最適とみられるオプションを選び続ける．

行列 msc01440, rail_1357, mhd3200_b, orgseg_1 に対しては提案手法の 2 つの変種 Pre-Ex と NIPD で regret に大きな差はなかったが，一部では stddev が改善した．また ex29, watt_2, lung1 では Pre-Ex よりも NIPD の regret が小さくなった．これらの行列の自動チューニング履歴を図 5 に示す．行列 ex29 と watt_2 では，Pre-Ex では一度収束しなかったオプションが他のオプションがすべて試されるまで使われなかったところ，NIPD では改善している．また lung1 では収束確率が低く初期実験だけでは情報が不十分なところ，NIPD では改善している．またいずれの行列でも単純な自動チューニングである naive よりも提案手法である NIPD が小さい regret を示した．特に $N = 50$ のときには ave は 20% から 55% 減少した．

5. まとめ

本研究では直野・佐川らにより提案された GA-GMRES を BiCG に適用する試みから，GA に関するオプションの最適化問題を導き，そのオンライン自動チューニング手法

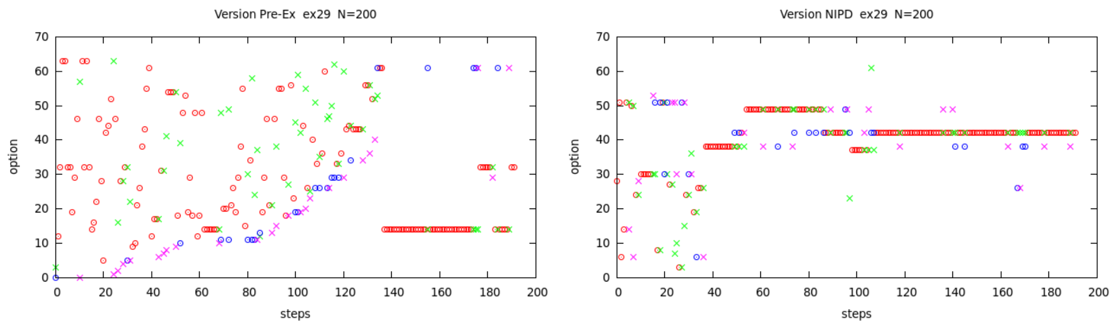
Vol.2017-HPC-159 No.2
2017/4/17

表 1 実験に用いた行列

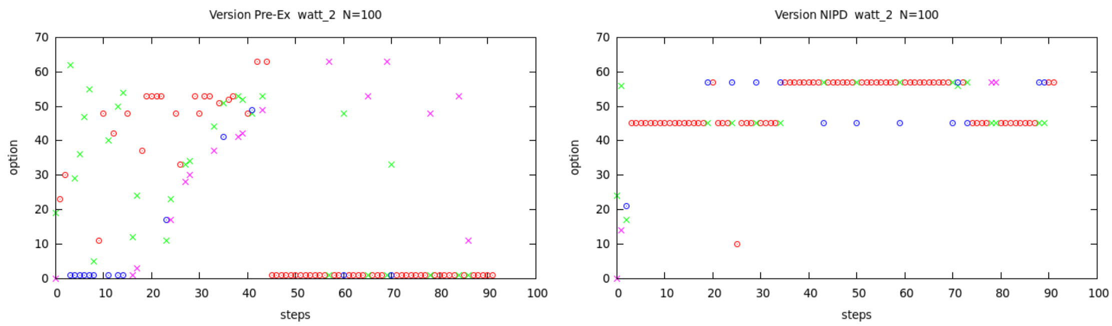
name	size	nonzeros	kind	option		p	μ	c
msc01440	1440	44998	structural	optimal	(50,1,12)	1.00	127	127
				pessimal	(200, 2, 1)	0.73	915	1679
rail_1357	1357	8895	model reduction	optimal	(50,3,4)	1.00	50	50
				pessimal	(100,2,1)	0.95	103	274
mhd3200b	3200	18316	electromagnetics	optimal	(50,3,12)	1.00	50.5	50.5
				pessimal	(200,2,1)	0.92	609	1226
ex29	2870	23754	fluid dynamics	optimal	(50,2,12)	0.92	169	729
				pessimal	(150,2,1)	0.21	855	6064
lung1	1650	7419	fluid dynamics	optimal	(50,0,12)	0.61	729	2125
				pessimal	(200,3,8)	0.02	2000	4245
wang2	2903	19093	semiconductor	optimal	(200,3,12)	1.00	698	704
				pessimal	(50,2,1)	0.77	1790	3114
orsreg_1	2205	14133	fluid dynamics	optimal	(200,3,4)	1.00	600	600
				pessimal	(50,1,1)	0.78	1408	2360
watt_2	1856	11550	fluid dynamics	optimal	(200,3,12)	0.87	1882	2257
				pessimal	(100,3,12)	0.02	2400	4778

図 5 チューニング履歴（Pre-Ex と NIPD）

ex29 N=200



watt_2 N=100



lung1 N=100

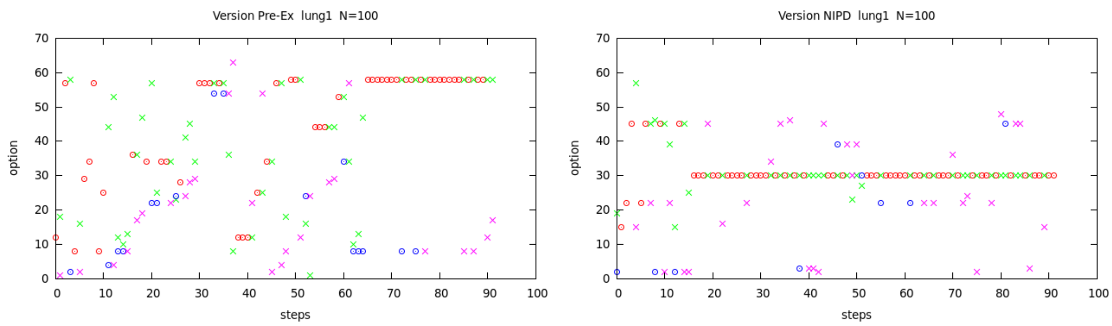


表 2　性能の比較 (1)

matrix	N	Version	ave	$stddev$	regret	loss
msc01440	50	Pre-Ex	197	9.6	70	2.4
		NIPD	190	9.1	**63**	5.3
		naive	424	19	297	731
	100	Pre-Ex	169	6.2	42	2.2
		NIPD	167	1.4	**40**	2.9
		naive	366	13	239	47
	200	Pre-Ex	152	2.5	**25**	3.6
		NIPD	152	1.8	**25**	1.9
		naive	267	19	140	55
rail_1357	50	Pre-Ex	68	7.1	18	108
		NIPD	67	6.1	**17**	24
		naive	108	3.7	58	150
	100	Pre-Ex	71	5.7	**21**	94
		NIPD	67	6.4	**21**	11
		naive	118	9.8	68	139
	200	Pre-Ex	59	3.5	9.1	73
		NIPD	58	1.1	**7.7**	13
		naive	103	3.2	53	134
mhd3200_b	50	Pre-Ex	88	17	**37**	2.1
		NIPD	89	15	38	5.7
		naive	191	30	140	215
	100	Pre-Ex	79	9.8	28	2.0
		NIPD	72	1.6	**22**	1.6
		naive	185	14	134	157
	200	Pre-Ex	73	7.5	22	0.72
		NIPD	68	6.6	**18**	5.0
		naive	151	10	101	85
ex29	50	Pre-Ex	1820	134	1090	1083
		NIPD	1515	106	**786**	703
		naive	2371	140	1641	1038
	100	Pre-Ex	1716	180	987	1118
		NIPD	1419	202	**690**	597
		naive	1930	124	1200	1095
	200	Pre-Ex	1686	45	956	551
		NIPD	1329	59	**599**	689
		naive	1757	301	1027	1226

表 3　性能の比較 (2)

matrix	N	Version	ave	$stddev$	regret	loss
lung1	50	Pre-Ex	2645	64	520	468
		NIPD	2326	69	201	411
		naive	3082	70	956	468
	100	Pre-Ex	2648	69	522	387
		NIPD	2334	73	**208**	362
		naive	2779	122	654	566
	200	Pre-Ex	2485	62	360	306
		NIPD	2321	119	196	336
		naive	2601	107	476	577
wang2	50	Pre-Ex	793	14	95	18
		NIPD	790	28	**92**	17
		naive	1256	56	558	23
	100	Pre-Ex	750	11	**52**	13
		NIPD	760	12	62	12
		naive	1067	25	369	99
	200	Pre-Ex	734	6.3	**36**	12
		NIPD	741	1.1	43	11
		naive	903	16	205	76
orsreg_1	50	Pre-Ex	627	9.0	**27**	0.33
		NIPD	635	10	35	12
		naive	792	5.2	192	1.2
	100	Pre-Ex	616	3.6	**16**	1.0
		NIPD	618	4.0	18	3.6
		naive	717	8.5	117	61
	200	Pre-Ex	610	3.0	10	0.14
		NIPD	609	1.0	**9.0**	3.0
		naive	691	15	91	59
watt_2	50	Pre-Ex	2972	108	715	301
		NIPD	2731	53	**474**	195
		naive	3722	48	1456	757
	100	Pre-Ex	2728	50	473	168
		NIPD	2557	50	**300**	152
		naive	3383	151	1126	906
	200	Pre-Ex	2680	76	423	151
		NIPD	2570	63	**313**	234
		naive	3084	231	827	700

を提案した．従来のオンライン自動チューニング手法であるワンステップ近似は正規分布を仮定していたため，反復法が収束しない場合に現れるコストが無限大という状況が適切に表現できなかったところ，収束確率の概念を導入してワンステップ近似を拡張することにより解決した．またSLHD に基づく少数の初期実験から効率よく性能を推定する手法を提案した．また分散および収束確率の推定方法を改善する手法を示した．これらの手法により効率的に性能のよいオプションを選ぶことができ，単純な自動チューニングに比べて最大約 2 倍の性能を達成した．

今後の課題としては，チューニング開始直後の挙動のさらなる安定化や，今回外れ値検出で補正した正規分布からの逸脱のより適切な処理が挙げられる．

謝辞　本研究の一部は，科研費 15H02708 と 15K12033 および JST CREST「進化的アプローチによる超並列複合システム向け開発環境の創出」の補助により実施された．

参考文献

[1] Naono, K., Zakaria, N., Sakurai, T., Pal, A. and Sagawa, N.: Evaluation of Genetic Algorithm on Initial Vector Settings for GMRES, *2013 SIAM Conference on Computational Science and Engineering*, Boston, Massachusetts (2013).

[2] 佐川暢俊：*A Study on Effective Generation and Solution of Matrices in the Process of Numerical Simulation*, 大阪大学 情報科学研究科 博士論文 (2015).

[3] Suda, R.: A Bayesian Method of Online Automatic Tuning, *Software Automatic Tuning: From Concepts to the State-of-the-Art Results* (Naono, K., Teranishi, K., Cavazos, J. and Suda, R., eds.), Springer, chapter 13, pp. 275–294 (2010).

[4] 須田礼仁：頑健で効率的なオンライン自動チューニングのための統計モデル，情報処理学会研究報告，Vol. 2008-HPC-116, pp. 109–114 (2008).

[5] Park, J.-S.: Optimal latin hypercube designs for computer experiments, *Journal of Statistical Planning and Inference*, Vol. 39, pp. 95–111 (1994).

[6] Welford, B. P.: Note on a method for calculating corrected sums of squares and products, *Technometrics*, Vol. 4(3), pp. 419–420 (1962).