オンライン自動チューニングのための Bayes 統計に基づく逐次実験計画法

須　田　礼　仁[†,††]

本論文は，ソフトウェア自動チューニングに Bayes 統計を用いることを提案する．具体的な手法として，本論文では multi-armed bandit problem に相当するオンライン自動チューニングの問題に対し， Bayes 統計に基づく逐次実験計画法を提案する．提案手法により，誤差を含むモデルからの情報と測定のばらつきを伴う実験データからの情報の融合が可能となり，効率的な自動チューニングが実現できることを，理論的考察と適用実験の両面から示す．

A Sequential Experimental Design based on Bayesian Statistics for Online Automatic Tuning

Reiji SUDA[†,††]

This paper proposes to use Bayesian statistics for software automatic tuning and a sequential experimental design based on Bayesian statistics for online automatic tuning which is equivalent to multi-armed bandit problem. From theoretical considerations and from experiments, we show that the proposed method successfully combines information from model with imprecision and that from empirical data with variance and realizes efficient automatic tuning.

1. はじめに

1.1 ソフトウェア自動チューニング

ソフトウェア開発において様々な選択肢からひとつを選ぶ，あるいはパラメタに値を設定するといった「決定」は常に必要である．決定のために実験が必要となることもあるが，実験の結果は実験条件に依存しているので，設定した実験条件が適切でなければ不適切な決定がなされてしまう可能性がある．あるいは実験条件に過適応したパラメタになっていて，一般的には不利な決定をしているという危険性もある．

これに対し，実運用条件下で実験を行い，それに従った決定を行う機構をソフトウェアの内部に備える**ソフトウェア自動チューニング**は，上記のような危険に対する一般性のある対策と考えることができる．ソフトウェアの自動チューニングは ATLAS[7] や FFTW[3] の劇的な成功を契機として広がりを見せている．なお，以下では「ソフトウェア自動チューニング」を単に「自動チューニング」と書く．

自動チューニングには統計が必要である．例えば所要時間などの測定ばらつきは無視できない問題であるが，自動チューニングの分野ではこれまで適切に扱われてきていないようである．例えば，「標本分散が一定の値以下になるまで実験を繰り返す」というような手法は統計学的には正当化されない．2 次モーメントである分散は平均以上にばらつきが大きく，標本分散がたまたま小さな値になった（早すぎる）時点で実験が打ち切られているはずである．また，網羅的な実験が非現実的な場合には，何らかのモデルから最適パラメタを推定するといった手法が必要になるが，ここでモデルのパラメタ値の推定において再び測定のばらつきが問題になる．このばらつきは繰り返し実験で抑えられるが，他方で実験コストを無視することもできない．これらの問題に関して統計学には実験計画法という分野があるのだが，自動チューニングの様々な応用場面を考えると，古典的手法だけでは十分とはいえない．

1.2 Bayes 統計に基づく自動チューニング手法

これに対し我々は，自動チューニングの基礎数理として Bayes 統計を用いることを提案する．Bayes 統計の特徴は，統計分布の未知パラメタ（例えば正規分布

† 東京大学 情報理工学系研究科 コンピュータ科学専攻
Dpt. of Computer Science, Grad. Schl. of Information Science and Technology, the University of Tokyo
†† 科学技術振興機構 CREST/JST, CREST

なら平均と分散）に関する情報の不確定性を，それらのパラメタがある確率分布に従うという形で表現することにある．これを**事前分布**と呼ぶ．そして事前分布と観測値を Bayes の定理により結合することで，**事後分布**と呼ばれる新たな情報を得る．我々は，性能予測モデルが与える情報を事前分布に対応させ，実際の観測値とあわせて事後分布を得ることにより，モデルと実測を合わせた情報をその不確定性の定量的な推定と共に求め，それに基づいて自動チューニングのアルゴリズムを設計することを提案する．

本論文では，具体的な手法として，Bayes 統計に基づくオンライン自動チューニングのための逐次実験計画を提案する．そして，この手法が，誤差を含むモデル，ばらつきのある観測という環境下でも，少ない実験コストで必要な情報を安定的に収集できることを，理論的考察と実験を通じて示す．以下，2 節では Bayes 統計の基礎を説明し，逐次実験計画法を提案し，その性質について理論的に考察する．3 節では行列積を例題として提案手法の評価を行う．4 節はまとめである．

なお，本論文は統計学の立場で手法の提案や解析をするものではない．本論文の主張は Bayes 統計を自動チューニングに適用することにより優れた手法が得られるという点にあり，評価も自動チューニングの視点で行う．本研究の初期段階の結果は iWAPT2007[5] で発表しているが，これは「使ってみたらうまくいくようだ」という段階での報告であり，(1) 逐次実験計画の性質の議論が厳密でない，(2) 事前情報の設定に統計学的に見て難がある，(3) t 分布を正規分布で近似しており提案手法を正確に実装していない，という問題点があった．本稿では (1) 提案する逐次実験計画の性質を数学的に定式化・証明した，(2) 事前情報を統計情報から導いている，(3) t 分布を正しく計算している，という点で改善されている．

最後に関連研究について言及する．著者の知る限り，Bayes 統計を自動チューニングに使う提案は他にはない．頻度統計に基づく興味深い手法が Vuduc ら[6] により提案されているが，モデルはなく実験だけで性能評価を行う点が本研究とは大きく異なる．

2. Bayes 統計に基づく逐次実験計画

実験計画とは，効率的に情報を得るためにどの実験をどの順序で行うかという問題を扱う統計学の一分野である．逐次実験計画とは，実験データを見てそれ以降の実験内容を決める実験計画で，少ない実験で効率的に情報を集められるという特長がある．しかし頻度統計の枠組みで逐次実験計画を論ずるのは難しく，Bayes 統計が用いられることが多い．

本節では，まず本稿で用いる Bayes モデルを紹介したあと，Bayes 統計における最適逐次実験計画を示し，それを近似した逐次実験計画を提案する．そして提案手法の性質を理論的な視点で考察する．

2.1 Bayes 統計の基礎と使用する統計モデル

Bayes 統計では，確率変数 y の期待値 μ はどうやら $\mu_0 \pm \tau$ ぐらいの範囲にありそうだという情報を，μ が（例えば）平均 μ_0, 分散 τ^2 の正規分布 $N(\mu_0, \tau^2)$ に従う，すなわち μ の確率密度分布が

$$p(\mu) = \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{(\mu-\mu_0)^2}{2\tau^2}\right)$$

である，のように表現する．もし y が分散 σ^2 が既知の正規分布 $N(\mu, \sigma^2)$ に従うとすると，与えられた μ に対して実測値 $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ が発生する確率は

$$p(\boldsymbol{y}|\mu) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{\sum_i (y_i-\mu)^2}{2\sigma^2}\right)$$

となる．ここで Bayes の定理

$$p(\mu|\boldsymbol{y})p(\boldsymbol{y}) = p(\boldsymbol{y},\mu) = p(\boldsymbol{y}|\mu)p(\mu)$$

を用いると

$$\begin{aligned} p(\mu|\boldsymbol{y}) &\propto \exp\left(-\frac{\sum_i (y_i-\mu)^2}{2\sigma^2} - \frac{(\mu-\mu_0)^2}{2\tau^2}\right) \\ &\propto \exp\left(-\frac{(\mu-\mu_n)^2}{2\tau_n^2}\right) \end{aligned}$$

となる (σ^2, τ^2, y_i, $p(\boldsymbol{y})$ は定数とみなす)．ここで

$$\begin{aligned} \tau_n^2 &= \frac{\sigma^2}{\kappa_n} \qquad \kappa_n = \frac{\sigma^2}{\tau^2} + n \\ \mu_n &= \frac{\kappa_0\mu_0 + n\bar{y}}{\kappa_0 + n} \qquad \bar{y} = \frac{1}{n}\sum_i y_i \end{aligned} \tag{1}$$

である．このとき $p(\mu)$ を**事前分布**, $p(\mu|\boldsymbol{y})$ を**事後分布**という．この事後分布を用いると，次の観測 y_{n+1} の値がどうなるか予測する分布が

$$p(y_{n+1}|\boldsymbol{y}) = \int p(y_{n+1}|\mu)p(\mu|\boldsymbol{y})d\mu$$

のように μ の分布に関する期待値を取ることで得られる．上の例でこの積分を実行すると，

$$\begin{aligned} y_{n+1}|\boldsymbol{y} &\sim N(\mu_n, \sigma_n^2) \\ \sigma_n^2 &= \sigma^2\kappa_{n+1}/\kappa_n \end{aligned}$$

とやはり正規分布となる．なお，$y \sim N(\mu, \sigma^2)$ は y が平均 μ, 分散 σ^2 の正規分布に従うことを表す．

上記のモデルを本論文では**分散既知の正規分布モデル**と呼ぶ．これに対し**分散未知の正規分布モデル**は，分散の事前分布を Inv-χ^2 分布

$$p(\sigma^2) = \frac{(\nu_0/2)^{\nu_0/2}}{\Gamma(\nu_0/2)} \frac{(\sigma_0^2)^{\nu_0/2}}{(\sigma^2)^{\nu_0/2+1}} \exp\left(-\frac{\nu_0\sigma_0^2}{2\sigma^2}\right)$$

としたうえで，平均の事前分布を

$$\mu|\sigma^2 \sim N(\mu_0, \sigma^2/\kappa_0)$$

とする．このとき y_{n+1} の予測分布は t 分布となり，期待値はやはり μ_n である．

これらのモデルは Bayes 統計でよく知られたもので，より詳しくは 2) のような教科書を参照されたい．

2.2 Bayes 最適逐次実験計画

Bayes 統計に従うと，逐次実験計画の最適解が以下のように求められる．まず問題を定式化する．これは multi-armed bandit problem として古くから知られているもの[1] で，下記の最適解も既知である．

あるプログラムがあるライブラリ関数を繰り返し呼び出す．ライブラリ関数には複数の実装があり，すべて機能は同等だが所要時間のみが異なる．但しどの実装が最も速いか事前には分からない．そこで，反復呼び出しの一部を性能評価に用いることにより，各実装の速度に関する情報を得る．目的は反復呼び出し全体の実行時間の最小化である．なお，性能評価に用いる呼び出しも有効な計算として利用する．また，呼び出し回数は事前に分かっている固定数とする．

最適解は最後の呼び出しから逆順に決めてゆく．以下では $\mu_i^{(k)}$ は最後から k 回目の呼び出しの直前での事前分布による実装 i の所要時間の期待値を表す．

もし実装 i を最後の呼び出しで使うとすると，所要時間の期待値は $\mu_i^{(1)}$ である．よって最後の呼び出しの所要時間を最小にするには $\mu_i^{(1)}$ を最小にする i を選べばよい．最適な選択をした場合の最後の呼び出しの所要時間の期待値は $\min_i\{\mu_i^{(1)}\}$ である．

次に最後から 2 回目の呼び出しについて考える．実装 i を最後から 2 回目に使い，最後の 1 回では最適な選択をしたときに，最後の 2 回の呼び出しにかかる所要時間の合計 $w_i^{(2)}$ を考える．最後から 2 回目の実行時間を y とすると，y により更新された事後期待値が $\mu_i^{(1)}$ となる．今回のモデルでは，実装 i がこれまでに実行された回数を $n-1$ とすると，式 (1) から

$$\mu_i^{(1)} = \frac{\kappa_{n-1}\mu_i^{(2)} + y}{\kappa_n}$$

となる．最後の呼び出しの所要時間は上記の考察から $\min_j\{\mu_j^{(1)}\}$ である．よって

$$w_i^{(2)} = \mu_i^{(2)} + E(\min_j\{\mu_j^{(1)}\})$$

となる．ここで $E(x)$ は x の期待値を表す．そして，$w_i^{(2)}$ を最小にする実装 i を最後から 2 回目に選択するのが最適となる．

同様に最後から k 回目について考える．最後から k 回目に実装 i を選択し，その後は最適な選択をした場合の，最後の k 回の呼び出しの所要時間の合計を $w_i^{(k)}$ とすると，

$$w_i^{(k)} = \mu_i^{(k)} + E(\min_j\{w_j^{(k-1)}\}) \tag{2}$$

となり，これを最小にする i を最後から k 回目に選択するのが最適である．$E(x)$ は一般に積分で表されるが，式 (2) の E の中の $w_j^{(k-1)}$ も再帰的に E を用いて定義されるため，$w_i^{(k)}$ は多次元積分となり，高精度に計算するのは一般的には困難である．このため最適解が得られるのはごく簡単な場合に限られる．

2.3 逐次実験計画法の提案

前節の実験計画は最適だが一般には計算が困難である．そこで本論文では最後の k 回の呼び出しの所要時間の期待値を次のように近似することを提案する．

$$\tilde{w}_i^{(k)} = \mu_i^{(k)} + (k-1)E(\min\{\mu_i^{(k-1)}, \mu_{\min'}^{(k)}\}) \tag{3}$$

ここで $\min'$ は実装 i 以外で所要時間の期待値が最小の実装を表す．実験計画としては，最後から k 回目の実行では $\tilde{w}_i^{(k)}$ を最小にする実装 i を実行する．

この $\tilde{w}_i^{(k)}$ は残りの $k-1$ 回の呼び出しにおいて「すべて実装 i」または「すべて実装 $\min'$」という 2 通りだけを考慮していることに相当する．最適解ではすべての実装のあらゆる実行を考慮して所要時間の期待値を最小化しているのに比べると相当に簡単であるが，以下に示すようにそれなりに有効である．

所要時間 y の予測分布を $p_i(y)$ とおくと，近似コスト式 (3) の具体的な計算は

$$\tilde{w}_i^{(k)} = \mu_i^{(k)} + (k-1)\int_{-\infty}^{\delta_i} \mu_i^{(k-1)} p_i(y)dy + (k-1)\int_{\delta_i}^{\infty} \mu_{\min'}^{(k)} p_i(y)dy$$

という 1 次元積分となる．ここで δ_i は $y=\delta_i$ のとき $\mu_i^{(k-1)} = \mu_{\min'}^{(k)}$ となるような値で，今回の例では

$$\delta_i = \kappa_n\mu_{\min'}^{(k)} - \kappa_{n-1}\mu_i^{(k)}$$

である．今回は p_i は正規分布または t 分布であるので，統計ライブラリを用いて容易に計算できる．

2.4 提案する実験計画手法の性質

ここでは提案する逐次実験計画法の性質について，やや一般化した仮定で理論的に考察する．まず Bayes モデルに対し，次の仮定をおく．

- 予測分布 $p_i(y)$ は連続微分可能で

  $$p_i(y) = \pi_i(|y - \mu_i^{(k)}|)$$

  の形に書ける．

- 事後平均 $\mu_i^{(k-1)}$ は実数 α_i により

$$\mu_i^{(k-1)} = \alpha_i y + (1-\alpha_i)\mu_i^{(k)}$$

の形に書ける．

両条件とも今回のモデルでは成立している．また，実装 i よりも実装 j のほうが**不確定性が大きい**ということを，上記の条件に現れる α_i, π_i に関して

$$\alpha_j > \alpha_i$$
$$\int_{-\infty}^{a} \pi_j(\eta)d\eta > \int_{-\infty}^{a} \pi_i(\eta)d\eta \quad (\forall a < 0)$$

が両方成立することと定義する．また，

$$\pi_i = \pi_j, \qquad \alpha_i = \alpha_j$$

が成り立つとき，**不確定性が等しい**ということにする．分散既知の正規分布モデルでは「大きな不確定性」は「少ない実験回数」におよそ対応するが，分散未知の正規分布モデルでは両者が一致しない場合も生じる．

以上の仮定のもとで，次のことが証明できる[8]．

- 実装 i と j の不確定性が等しいとき，$\mu_i^{(k)} < \mu_j^{(k)}$ なら，$\tilde{w}_i^{(k)} < \tilde{w}_j^{(k)}$ となる．
- 実装 i よりも j の不確定性が大きいとき
  - $\mu_j^{(k)} \le \mu_i^{(k)}$ なら，$\tilde{w}_j^{(k)} < \tilde{w}_i^{(k)}$ となる．
  - $\mu_j^{(k)} > \mu_i^{(k)}$ であっても，$\mu_j^{(k)}$ が $\mu_i^{(k)}$ に十分近ければやはり $\tilde{w}_j^{(k)} < \tilde{w}_i^{(k)}$ となる．
  - $\mu_j^{(k)}$ が $\mu_i^{(k)}$ より十分大きくなると $\tilde{w}_j^{(k)} > \tilde{w}_i^{(k)}$ となる．

これを $\tilde{w}_i^{(k)}$ を最小にするものが実行されることを考慮して言い換えると次のようになる．

- 所要時間の期待値が大きくても，不確定性が大きい場合には，実行される可能性がある．
- 逆に所要時間の期待値が大きく，不確定性が小さい場合には，実行される可能性はない．

すなわち，単に速い実装を選択するのではなく，推定所要時間と情報の不確定性を考慮し，情報を集めるため遅い実装を選択するということもありうるのである．また，下に見るように，残り呼び出し回数 k が大きいほど情報を集める方の優先度が高い．これは残り呼び出し回数（将来の利用回数）が多いほど，実験コストをかけて最適化する価値があるという直感に一致する．

以下では $\mu_0^{(k)} \le \mu_1^{(k)}$ を満たす 2 つの実装があると仮定して具体的な計算を行い，上述の性質を確認する．図 1 の左図は，分散 $\sigma^2 = 10.0$ が既知の正規分布モデルで $\mu_0^{(2)} = 0.0$, $\kappa_0 = 1.5$ としたときの $\tilde{w}_0^{(2)}$ と $\tilde{w}_1^{(2)}$ を $\mu_1^{(2)}$ の関数として表したものである．実線は $\tilde{w}_0^{(2)}$, 破線は $\tilde{w}_1^{(2)}$ であり，それぞれ 4 つあるプロットは上から順にこれまでの実験回数 n が 3, 2, 1, 0 の場合にそれぞれ対応している．すなわち，観測数 n が大きくなり不確定性が下がると，$\tilde{w}_0^{(2)}$ も $\tilde{w}_1^{(2)}$ も関数値が大きくなっている．値が最小のものが選択されることから，観測数が少なく不確定性が高いものほど選ばれやすいことが確認できる．また，$\tilde{w}_1^{(2)}$ の各曲線は単調増加であることから，観測数（不確定性）が同じであれば所要時間の短い方が必ず選ばれることがわかる．さらに，実装 0 と 1 で n が同じ場合は，$\mu_1^{(2)} = \mu_0^{(2)}$ $(= 0)$ では $\tilde{w}_0^{(2)} = \tilde{w}_1^{(2)}$ であり，$\mu_1^{(2)} > 0$ では $\tilde{w}_0^{(2)} < \tilde{w}_1^{(2)}$ となっている．これも不確定性が等しければ所要時間の期待値が小さい方が選択されることを意味している．しかし，実装 0 の観測数が実装 1 の観測数より多い場合は $\mu_1^{(2)}$ が 0 に近いところで $\tilde{w}_0^{(2)} > \tilde{w}_1^{(2)}$ となる部分があり，不確定性が大きい場合には所要時間の期待値が大きい実装が選択されうることがわかる．図 1 の右図は同じ条件で $k = 3$ の場合を示している．ここでは観測数 n ごとの関数値の差が大きくなっている．例えば $n = 1$ の $\tilde{w}_0^{(2)}$ と $n = 0$ の $\tilde{w}_1^{(2)}$ の交点は $\mu_1^{(2)} = 0.4$ 付近であるが，$k = 3$ での交点は $\mu_1^{(3)} = 0.7$ 付近にあることから，$k = 2$ に比べ $k = 3$ では不確定性が大きいものがより選ばれやすくなっていることがわかる．

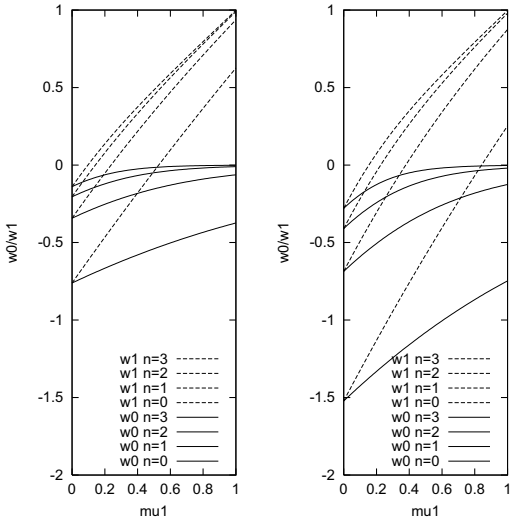

図 1 推定コスト $\tilde{w}_0^{(k)}$ と $\tilde{w}_1^{(k)}$．左：$k = 2$, 右：$k = 3$
Fig. 1 Estimated costs $\tilde{w}_0^{(k)}$ and $\tilde{w}_1^{(k)}$; left: $k = 2$, right: $k = 3$

3. 行列積ルーチンへの適用と評価

この節では，提案手法によるオンライン自動チューニングの例を示す．例題として行列積を挙げるが，以下の行列積の実装はあくまで逐次実験計画の有効性を

示すためのものであり，各マシンでぎりぎりの性能を追求するものではないことにご注意いただきたい．

3.1 問題設定

例題は $N \times N$ の正方行列 B, C の積

$$A = BC$$

を計算するプログラムである．いわゆる内積形式の単純なプログラムを，片桐らにより開発された ABCLibScript[4] により外側・中間の 2 つのループをアンロールした．外側ループのアンロール段数を u_i, 中間ループのアンロール段数を u_j としたとき，

$$\begin{aligned}(u_i, u_j) \in\ & [1,128] \times [1,1] \ \cup\ [1,64] \times [1,2] \\ & \cup\ [1,32] \times [1,4] \ \cup\ [1,16] \times [1,8] \\ & \cup\ [1,8] \times [1,16] \ \cup\ [1,4] \times [1,32] \\ & \cup\ [1,2] \times [1,64] \ \cup\ [1,1] \times [1,128]\end{aligned}$$

という範囲でアンロールを行った．これにより，アンロールされていないコードも含めて 576 通りの異なるサブルーチンが得られた．

オンライン自動チューニングの問題設定としては次のようになる．プログラムが各行列サイズ N に対して M 回ずつ行列積ルーチンを呼び出す．上記の 576 のサブルーチンを合計 M 回呼び出し，その合計所要時間を最小化したい．以下では N は 8 から 256 または 512 まで (上限はマシンの速度による)，M は基本的に 1000 とした．

3.2 性能モデル

アンローリングが性能に与える影響を正確にモデル化するのは容易ではない．しかし，Bayes 統計ではモデルがある程度不正確であっても，それなりの情報を引き出すことができる．このことの例証のため，今回は次のような簡単な性能モデルを構築した．

ループをアンロールすると，アンロール段数で割り切れる「商」のループと，割り切れない「剰余」のループが生じる．反復回数を N, アンロール段数を u とすると，商ループは N/u 回 (但し整数除算の商を表すものとする，以下同様)，剰余ループは $N \bmod u$ 回まわる．これを用いれば，コードの各部分でロード・ストア・演算などの命令が何回行われるか計算することができる．さらにループ制御の命令などを反復回数の定数倍で見込む． ABCLibScript がアンロールしたコードをこのように分析すると，以下の 15 項が得られる (演算回数：演算内容の形で示す)．なお，外側の商ループを「外商」などのように略記する．

- 1: サブルーチン呼び出し
- N/u_i: 外商ループの制御
- $(N/u_i)(N/u_j)$: 外商・中商ループの制御
- $(N/u_i)(N/u_j)u_iu_j$: 外商・中商の A のロード・ストア
- $(N/u_i)(N/u_j)N$: 外商・中商・最内ループの制御
- $(N/u_i)(N/u_j)Nu_i$: 外商・中商の B のロード
- $(N/u_i)(N/u_j)Nu_j$: 外商・中商の C のロード
- $(N/u_i)(N/u_j)Nu_iu_j$: 外商・中商の積和演算
- $(N/u_i)(N \bmod u_j)$: 外商・中剰余ループの制御
- $(N/u_i)(N \bmod u_j)u_i$: 外商・中剰余の A のロード・ストア
- $(N/u_i)(N \bmod u_j)N$: 外商・中剰余・最内ループの制御, C のロード
- $(N/u_i)(N \bmod u_j)Nu_i$: 外商・中剰余の B のロード，積和演算
- $(N \bmod u_i)$: 外剰余ループの制御
- $(N \bmod u_i)N$: 外剰余・中ループの制御，A のロード・ストア
- $(N \bmod u_i)N^2$: 外剰余・中・最内ループの制御，B と C のロード，積和演算

あとは演算などの所要時間がわかれば所要時間が推定できるが，以下では最小二乗法（速いものほど当てはめがよくなるように所要時間の逆数を重みとした）で推定している．但し，推定しているのは命令ごとの時間ではなく，上記 15 項それぞれの定数係数である．

このモデルには，アンロールしすぎによるレジスタ退避コードや，行列サイズに依存するキャッシュミスによる性能変化などが含まれていない．今回は，前者はモデルの誤差として扱い，後者は N ごとに係数を決定することで対応することとする．N ごとに係数を決めることとしたため，上記の 15 項は線形独立にはならず，階数は 7 となる．従って，係数を決定するのに必要な実験数も 7 となる．

本モデルのあてはめの良し悪しはマシンやコンパイラに依存するが，以下の実験では比較的あてはめがよい場合が多かったようである．

3.3 事前分布

事前分布は Bayes 統計の強みでもあり，弱みでもある．適切な事前分布を与えれば Bayes 統計は極めて強力であるが，事前分布が不適切であれば Bayes 統計はまったく信頼できない結果を与える．最適逐次実験計画 (2) が最適なのは，事前分布が「正しい」場合である．すなわち，事前分布に従ってパラメタを決めた多数の問題を生成すれば，その平均性能が他の方法よりもよい．今回は行列サイズ N ごとに実験を行いその平均を取っているので，多数の問題の平均という状況にやや近い．しかし，具体的な一つ一つの例については，他の手法に比べて良い場合も悪い場合もある（Stein のパラドックスと根は同じである）．

事前分布の構成法にはいろいろあるが，最適逐次実験計画 (2) は事前分布が最初に与えられ変更されないことを仮定している．そこで本稿では事前サンプリングによる統計的推定で事前分布を構成した．

今回用いた具体的な事前分布は次の通りである．分散 σ^2 については，行列サイズ N ごとに $\log\sigma^2$ の平均を取り，$\log N$ に関する区分線形関数で近似した．分散未知モデルで必要となる ν_0 は N ごとにモーメント法で求め，$\log(\nu_0-4)$ の平均から得られる値をすべての N で共通に使った．平均の期待値 μ_0 は前節のモデルによる推定値をそのまま用いた．モデルの重み κ_0 は，モデルの二乗誤差の相乗平均 τ^2 と分散 σ^2 から $\kappa_0=\sigma^2/\tau^2$ で計算し，$\log\kappa_0$ を $\log N$ に関する区分線形関数で近似した．これらの手法には再検討の余地が残っているが，今回はいくつか試みたところ結果が比較的よかったものという報告にとどめておく．

3.4 比較対象と比較方法

以下の実験では，次の 4 つの手法を比較する．

- 手法 M_1: 最初の 576 回ですべての実装を 1 回ずつ実行し，その後は平均所要時間が最短のものを選択する．平均所要時間はその実装のすべての実行に関する平均である．
- 手法 M_2: 最初の 7 回でモデルを構築し，その後はモデルが最速と推定するものを選択する．但し，実際に実行したことがある実装の推定実行時間はモデルではなく，実測平均とする．
- 手法 M_{3A}, M_{3B}: 最初の 7 回でモデルを構築し，その後は本論文が提案する逐次実験計画法 (3) により選択する．M_{3A} は分散既知の正規分布モデル，M_{3B} は分散未知の正規分布モデルによる．

最適実験計画 (2) との比較が望まれるが，膨大な数の数百次元無限区間積分を精度よく計算する必要があり，実現が容易でないので今回はやむをえず見送る．

図 2 に呼び出し回数が 1000 回の場合の手法 M_1, M_2, M_{3B} による実験の様子を示す（M_{3A} は M_{3B} とよく似ているので省略）．横軸に呼び出し回数，縦軸に選択された実装の平均実行時間（最速実装との相対値）を示す．M_1 では最初すべての実装を試し，その後は最も速かったものを選択する．M_2 では最初の 7 回でモデルを構築し，その後はモデルが最速と推定するものを選択する．M_{3B} ではモデルを参照しながら実験も行い，M_1 より少ない実験数で M_2 よりよい実装を選択している．なお，マシンは以下の実験で最もモデルのあてはめが悪い Core2Duo 2.3GHz である．

実験に用いるデータは次のように生成した．まず，各ルーチンの実行時間を行列サイズごとに最低 10 回計測した．これにはマシンによっては 1 日以上かかるが，それでも以下の実験には不足なので，実験データから Bayes 予測法で人工的なデータを生成し，実データに追加・シャッフルして実験に用いた．以下ではデータ生成とシャッフルに用いる乱数の種を変化させて 5 回ずつ実験し，評価値の平均値を報告する．乱数の種は規則的に変化させているので，3 つの手法に対して同一のデータセットで実験が行われている．なお，所要時間は行列積ルーチンの所要時間であり，最小二乗法や積分など実験計画のための計算の時間は含まれていない．これについては本節の最後に考察する．

評価に用いる指標は以下の 2 つである．いずれも，行列サイズ N ごとに実験と評価を行い，その平均値を指標として用いる．

第 1 の指標は**相対リグレット** rr であり，

$$rr=(T_M-T_{opt})/T_{opt}$$

で定義する．ここで T_M は手法 M による合計所要時間，T_{opt} は最速ルーチンの平均所要時間に呼び出し回数をかけたものである．最適逐次実験計画はこれを最小にするものなので，提案手法が期待どおりの動作をしているかどうかを判定する主要な指標となる．なお，最速ルーチンが事前に分かっていなければ $rr=0$ は不可能である．

第 2 の指標は**相対ロス** rl であり，

$$rl=(t_M-t_{opt})/t_{opt}$$

で定義する．ここで t_M は手法 M がもっとも多く選択したルーチンの平均所要時間，t_{opt} は最速ルーチンの平均所要時間である．多くの実験計画ではこれを最小にしようとするのだが，今回は実験コストとのトレードオフで犠牲になる．但し，呼び出し回数が多くなるに従って実験コストは相対的に小さくなるため，rl が 0 に近づくことが望ましい．手法が最速ルーチンを常に検出できれば $rl=0$ となる．

従来の自動チューニングではロスが主な目的関数で，リグレットは効率を表す副次的な指標であった．しかし，あまり利用しないルーチンに時間をかけて最適化するのは適切だろうか．本論文では，最適化と利用の合計コストを総合的に最小化することがユーザにとって益であると想定し，リグレットを主な指標とした．

3.5 実験結果

図 3 に，呼び出し回数が 1000 回の場合について 4 つの手法による相対リグレットの値をいくつかのマシンで評価したものを示す．多様な環境で評価がしたいので，コンパイラも計時方法もあえて統一していない．マシンは CPU とクロック周波数 (GHz) で示し，M_2 の相対リグレットの順に並べた．

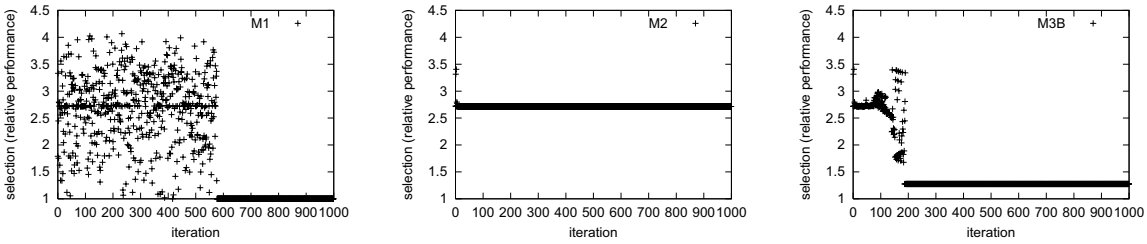

図 2 手法 M_1, M_2, M_{3B} による実行の様子
Fig. 2 Executions by M_1, M_2 and M_{3B}

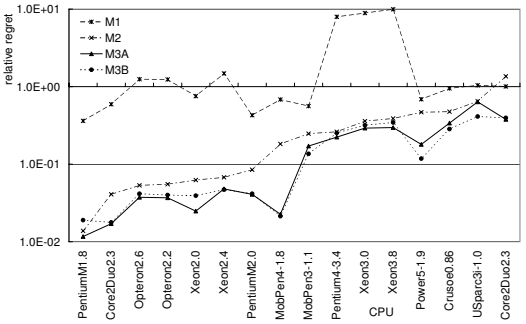

図 3 各手法の相対リグレットの評価．呼び出し回数 1000 回
Fig. 3 Relative regrets of the methods; 1000 iterations

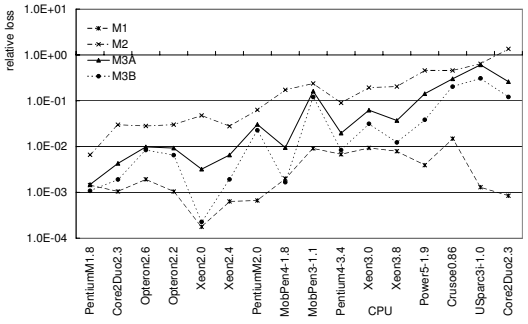

図 4 各手法の相対ロスの評価．呼び出し回数 1000 回
Fig. 4 Relative losses of the methods; 1000 iterations

図 3 を見ると，ほとんどのマシンで M_{3A} と M_{3B} は M_1 と M_2 のいずれよりも小さい相対リグレットを達成している．M_{3A} と M_{3B} の結果はかなり近いが，モデルの精度がよい左端の方では分散既知の M_{3A} が，モデルの精度が悪い右端の方では分散未知の M_{3B} がわずかによい．分散既知よりも分散未知の方が不確定性を大きく見積もり実験が増えるが，モデルの精度がよいとこれが無駄になり，モデルの精度が悪いと役立つものと思われる．

M_1 と M_2 の比較では，多くの場合で M_2 の方が小さい相対リグレットを得ている．しかし右端の Core2Duo では M_2 の相対リグレットが極端に大きい．詳細な分析により，最速に近い実装に対するモデルのあてはめが極端に悪いことが分かっている．これはモデルを用いる手法には極めて不利であるが，それでも M_{3A}, M_{3B} は M_1 よりも小さい相対リグレットを達成しており，提案手法が誤差の大きなモデルからも安定して情報を獲得していることがわかる．

図 4 は相対ロスを図 3 と同様にプロットしたものである．ほとんどのマシンで，相対ロスが少ないものから M_1, M_{3B}, M_{3A}, M_2 の順になっているが，これはおよそ実験回数の順序である．ときおり M_{3B} の相対ロスの方が M_1 よりも少なくなっているが，これは M_1 が一度きりの測定に頼っているためである．

表 1 呼び出し回数を変えたときの評価．Core2Duo 2.3GHz
Table 1 Evaluations with different numbers of iterations on Core2Duo 2.3GHz

指標	呼び出し回数	M_1	M_2	M_{3A}	M_{3B}
rr	100	—	1.405	1.11	1.11
	1000	1.006	1.358	0.40	0.38
	10000	0.101	1.353	0.19	0.13
rl	100	—	1.352	0.83	0.80
	1000	0.0008	1.353	0.26	0.12
	10000	0.0005	1.353	0.17	0.06

次に呼び出し回数を 100, 1000, 10000 と変化させて比較する．10000 回の実験には非常に時間がかかるため，Core2Duo 2.3GHz と Pentium4 3.4GHz のマシンのみで実験を行うこととした．表 1, 2 に結果を記す．有効数字は乱数の種を変えた 5 回の実験の標準偏差におよそ対応している．提案手法 M_{3A}, M_{3B} はともに呼び出し回数が増えるに従い rl が順調に減少してゆく．すなわち，呼び出し回数が多いほど実験を増やして最適に近い解を得るという性質が確認できる．これに対し M_1, M_2 は rl がほとんど変化しない．なお，全実装を一度ずつ試行する M_1 は呼び出し 100 回ではまだ試行の途中なので評価外とした．

最後に提案手法のための計算時間に関して報告する．

表 2 呼び出し回数を変えたときの評価．Pentium4 3.4GHz
Table 2 Evaluations with different numbers of iterations on Pentium4 3.4GHz

指標	呼び出し回数	M_1	M_2	M_{3A}	M_{3B}
rr	100	—	1.8	1.8	1.8
	1000	7.96	0.26	0.253	0.22
	10000	0.80	0.11	0.04	0.05
rl	100	—	0.09	0.04	0.03
	1000	0.007	0.09	0.02	0.01
	10000	0.006	0.09	0.01	0.002

表 3 各制御手法の 1 呼び出しあたりの計算時間
Table 3 Computational times of the control methods per iteration

マシン	M_1	M_2	M_{3A}	M_{3B}
Core2Duo	1.67e-6	1.12e-4	4.69e-4	1.90e-3
Pentium4	2.02e-6	1.36e-4	5.73e-4	2.47e-3

表 4 M_{3A} / M_{3B} が M_2 よりも有利になる行列サイズ
Table 4 Matrix sizes where M_{3A} or M_{3B} is advantageous over M_2

比較	M_2 vs M_{3A}		M_2 vs M_{3B}	
呼び出し回数	1000	10000	1000	10000
Core2Duo	65	60	110	100
Pentium4	320	145	310	270

但し現状の実装には改良の余地が多く，工夫次第で一桁以上の高速化が予想されるので，予備評価的な意味しかないことをお断りしておく．統計ライブラリのインストールの都合で先と同じ 2 台について評価した．

表 3 に各手法の 1 呼び出しあたりの平均計算時間を示す．Pentium4 3.4GHz の場合，呼び出し 1000 回での M_{3A} と M_2 の相対リグレットの差は表 2 よりわずか 0.007 なので，手法の計算を含めた所要時間で M_{3A} が M_2 に勝るためには，行列積の所要時間が約 0.06 秒以上でなければならない ($0.06 \times 0.007 =$ 4.2e-4 $\approx$ 5.73e-4 − 1.36e-4)．これは $N \approx 320$ 程度に相当する．このような計算の結果をまとめたものが表 4 である．これによれば，提案手法が有効となる行列サイズはマシンや呼び出し回数にかなり依存するが，いずれにせよ提案手法の計算時間は実用性を著しく損なうものではないことがわかる．

4. ま と め

本論文は自動チューニングに Bayes 統計を用いること，および，具体的な手法として，オンライン自動チューニングのための逐次実験計画法を提案し，理論的な考察と実験評価により提案手法の有効性を示した．Bayes 統計は不正確さを含むモデルとばらつきを含む実験データを結びつける数理的な手法であり，自動チューニングの基盤数理に適している．

今後の課題は多岐にわたるが，最も難しくまた興味深い問題が事前分布の構成である．今回はサンプリングにより事前分布を構成したが，事前サンプリングで得られた情報が事前分布の構築以外に使用されていない点が不自然である．また事前サンプリングのコストも考慮されていない．経験 Bayes や階層 Bayes など実験データから事前分布を更新する方法が解決の候補となるが，実験計画に与える影響については慎重に考慮する必要がある．また，今回は離散的な選択肢であったが，連続パラメタの最適設定も重要なテーマである．今後，これらの問題に精力的に取り組んでゆきたい．

謝辞

いつも貴重なコメントを頂いております東京大学の竹村彰通教授と自動チューニング研究会のメンバーの皆様に感謝いたします．

本研究の一部は文部科学省科学研究費「情報爆発時代のロバストな自動チューニングシステムに向けた数理的基盤技術の研究」および科学技術振興機構 CREST「ULP-HPC: 次世代テクノロジのモデル化・最適化による超低消費電力ハイパフォーマンスコンピューティング」の支援を受けています．

参 考 文 献

1) Berry, D.A., and Fristedt, B.: *Bandit Problem*, Chapman and Hall (1985).
2) Carlin, B. P. and Louis, T. A.: *Bayes and Empirical Bayes Methods for Data Analysis*, 2nd ed., Chapman and Hall (2000).
3) Frigo, M. and Johnson, S.G.: FFTW: an adaptive software architecture for the FFT, *Proc. ICASSP '98*, Vol. 3, pp. 1381–1384 (1998).
4) Katagiri, T., et. al: ABCLibScript: A directive to support specification of an auto-tuning facility for numerical software, *Parallel Computing*, Vol. 32, No. 1, pp. 92–112 (2006).
5) Suda, R.: A Bayesian Method for Online Code Selection: Toward Efficient and Robust Methods of Automatic Tuning, *Proc. 2nd Int'l Workshop on Automatic Performance Tuning (iWAPT2007)*, pp. 23–32 (2007).
6) Vuduc, R., Demmel, J. W., and Bilmes, J. A.: Statistical models for empirical search-based performance tuning, *Int'l J. High Perf. Comp. Appl.*, Vol. 18, No. 1, pp. 65–94 (2004).
7) Whaley, R. C. and Dongarra, J. J.: Automatically Tuned Linear Algebra Software, *Proc. SC98* (CD-ROM), (1998).
8) http://olab.is.s.u-tokyo.ac.jp/~reiji/proof.pdf