Vol.2010-HPC-126 No.41
2010/8/5

並列ソフトウェアのオンライン自動チューニングのための Bayes 的手法

須　田　礼　仁[†1]

本報告では並列ソフトウェアの局所的チューニングパラメタのオンライン自動チューニングのための Bayes 統計に基づく数理手法について論じる．まず，並列ソフトウェアの自動チューニングという問題において，無関連タスク問題，同一タスク問題，相関タスク問題という 3 種類の問題を提案する．本報告では，無関連タスク問題と同一タスク問題に関して Bayes 統計に基づく数理手法を論じる．特に同一タスク問題で，事前情報において候補の優劣の情報がない問題に対する手法を詳述する．本手法は，候補の数が無限であっても適用できるという特徴がある．シミュレーションにより評価した結果，従来手法に比べて Bayes 統計に基づく手法が優れていることが示された．

Bayesian Methods of Online Automatic Tuning of Parallel Software

Reiji Suda[†1]

This paper discusses mathematical methods based on Bayesian statistics for online automatic tuning of local tuning parameters of parallel software. First, three classes of problems of automatic tuning of parallel software are proposed: unrelated task problem, identical task problem, and correlated task problem. This paper proposes Bayesian methods for unrelated task problem and identical task problem. Especially, a method is discussed in detail for identical task problem without any a priori information about the performance difference of candidates. The proposed method is applicable to the problems with infinitely many candidates. Evaluation through simulations shows that the proposed Bayesian method outperforms an existing method.

†1 東京大学情報理工学系研究科 / JST CREST
Graduate School of Information Science and Technology, the University of Tokyo / JST CREST

1. はじめに

自動チューニングは，ソフトウェア自身が実行環境に対する適応能力を持つことにより，多様な条件において優れた性能を発揮することを目指す技術的パラダイムである．それを実現するための基本的なアイデアとして，ソフトウェアが可変性（**チューニングパラメタ**）を内蔵することと，このチューニングパラメタをどのように制御すれば最大の性能が発揮できるかを知るために実環境において性能評価を行うことの 2 点が挙げられる．このようなことから，自動チューニングの研究は，

(1) （**チューニング手法**）ソフトウェアにどのような可変性を実装すると多様な条件で優れた性能が発揮できるのか
(2) （**プログラミング**）そのような可変性を持つプログラムをどのように記述するか，記述しやすくするか
(3) （**システム**）性能試験やソフトウェアの実行制御のために必要なミドルウェアやシステムソフトウェアはどのようなものか
(4) （**数理**）最小限の性能評価から，最大限の性能を達成する情報を得るにはどうすればよいか

といった分野にわたる．（自動チューニングの問題設定について本稿では上記のような略述に留める．前稿[1] において議論したので参照されたい．）

著者はこれまで特に数理の部分に注目して研究を進めてきた．まず，逐次処理に対して Bayes 統計に基づくデータ解析・性能モデル化・逐次実験計画を提案してきた[1)–3)]．また，並列処理に関しては Bayes 統計に基づかない実験計画を提案してきた[4)–6)]．本稿は 7) に続き，並列処理のための Bayes 統計に基づく実験計画について論ずる．

本論に入る前に，我々の自動チューニングの数理モデルに関して必要な点を略述する．我々は，**チューニングパラメタ**と呼ばれる変数に適切な値を代入することにより，ソフトウェアの可変性・適応性が実現されると仮定する．チューニングパラメタは数値のこともあるが，アルゴリズムの選択やコード変換の有無など離散値・名目値のことも多い．チューニングパラメタに代入できる値を**候補**と呼ぶ．本稿では候補の集合は離散的で有限であるとする．自動チューニングでは実際の計算環境において，適当な候補を選択してソフトウェアを実行し，その性能を測定して，どの候補がよい性能を示すかを評価する．性能評価のためだけにソフトウェアを実行する**試行**と，実際にソフトウェアが使われる場面である**実施**という 2 種類の実行が考えられる．**オフライン自動チューニング**では，ソフトウェアの開発時やイン

Vol.2010-HPC-126 No.41
2010/8/5

ストール時に試行を行ってチューニングパラメタを選択し，実施時には性能評価もチューニングも行わない．他方**オンライン自動チューニング**では，一切試行を行わず，実施時に性能評価を行うことにより，優れた性能を示す候補を探す．本稿で論ずるのはオンライン自動チューニングである．また，性能指標には所要時間を用いるが，消費エネルギーでも同様の議論が可能である．

2. 並列処理における自動チューニング

本節では並列処理における自動チューニングについて，主に 6) で論じた**並列実験**について確認する．並列処理における自動チューニングのさまざまな可能性については 4) で概略を論じた．また，本稿では論じない**並列試行**に関しては 5) で基本的な結果を示している．

2.1 並列オンライン自動チューニング問題の定式化

並列処理は多数のプロセッサが処理に加わることにより，大規模な計算を短時間で実現することを目指すものである．並列処理の方式は現在複雑性を増しており，マルチコア SMP ノードにおける共有メモリ型のスレッド並列性と複数ノードにまたがる分散メモリ型の SPMD 並列性とを併用するハイブリッド並列化が広く行われている．また，GPU などにおける大規模（ベクトル長の長い）SIMD 並列性も，その高い性能からごく標準的に使われるようになってきている．本稿では MIMD 型の並列性を想定し，逐次処理を行うそれぞれの主体を**プロセッサ**と呼ぶことにする．(ハードウェア的な響きがあるが，ここでの「プロセッサ」はソフトウェア的な主体である．「プロセス」あるいは「スレッド」と呼んでもよい．)

並列処理におけるチューニングパラメタには，**大域的チューニングパラメタ**と**局所的チューニングパラメタ**とが考えられる．大域的チューニングパラメタは，ひとつのパラメタで並列処理全体の挙動を制御する．パラメタがひとつしかないので，候補を代入して性能を評価するということは，一度にひとつずつしかできない．このため，大域的チューニングパラメタの最適化は逐次処理の場合と同じ自動チューニングの数理手法を用いることになる．これに対し，局所的チューニングパラメタは，それぞれのプロセッサがひとつずつチューニングパラメタを持ち，それは当該のプロセッサの挙動にのみ影響を与える．パラメタが複数あるので，多数のプロセッサが異なる候補を選択して実行するということが可能となる．このため，局所的パラメタの制御には並列処理独特のチューニング手法が存在する．本稿で考察するのは，局所的チューニングパラメタの自動チューニングである．

本稿では，**図 1** のような簡単な並列処理を仮定する．P 個のプロセッサが協調動作をしてひとつの処理を実現する．処理全体は同一計算からなる K 回の**反復**で構成されている．

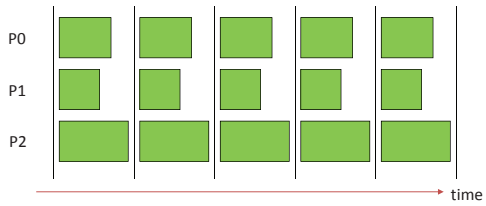

図 1 本稿で仮定する並列処理のモデル：長方形が計算を，縦線が同期通信を表す．
Fig. 1 Iterative computations in parallel processing — the rectangles represent computations, and the vertical bars represent synchronous communications.

本稿では，各反復は同一の計算から構成されるとする．また各反復は，プロセッサごとの独立な計算と，全プロセッサが関与する同期的な通信から構成されるとする．方程式の反復解法などはこのような並列処理となることが多い．本稿では，同期的な通信の部分についてはチューニングの対象としない．計算部分に影響する局所的チューニングパラメタがあると仮定し，これを制御することにより実行を最適化する．本稿で論ずるのはオンライン自動チューニングであり，**目的関数**は K 回の反復の合計実行時間である．通信部分はチューニングの対象ではないので考慮から外すことができるので，各反復の実行時間は，プロセッサごとの計算時間の最大値と定義する．

より厳密に定義しよう．$p \in \{1, 2, \ldots, P\}$ がプロセッサのひとつを表すとする．候補は M 個あるとし，$m \in \{1, 2, \ldots, M\}$ が候補のいずれかを表す．プロセッサ p において候補 m を採用して実行したときの所要時間を t_{pm} とあらわす．本稿では，簡単のため，実行時間のばらつきは無視できるとする．プロセッサ p が第 k 反復の実行において選択する候補を $m(p,k)$ とすると，第 k 反復の実行時間 T_k は

$$T_k = \max_{p=1}^{P} \{t_{pm(p,k)}\}$$

となる．これより，目的関数は

$$T = \sum_{k=1}^{K} \max_{p=1}^{P} \{t_{pm(p,k)}\}$$

であり，これを最小にするように $m(p,k)$ を定めたい．一般に $m(p,k)$ を選ぶ方法は**戦略**と呼ばれる．一般的に，$m(p,k)$ の決定においては，1 回目から $k-1$ 回目までの実行における所要時間

Vol.2010-HPC-126 No.41
2010/8/5

$t_{pm(p,j)} \quad p = 1, 2, \ldots, P \quad j = 1, 2, \ldots, k-1$

を参照することができる．

2.2 既存の戦略

ここでは 6) に述べられている 3 つのオンライン自動チューニングの戦略を述べる．

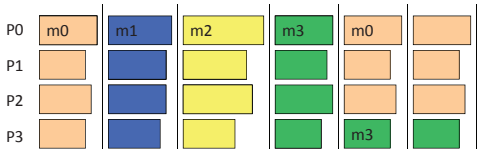

図 2 SEO
Fig. 2 SEO

第 1 の戦略は **SEO** (Serial Experiments Once) である（**図 2**）．この戦略では，最初の M 回の反復において，各プロセッサがすべての候補 $m = 1, 2, \ldots, M$ を試して計算時間 $(t_{pm})_{m=1}^{M}$ を得る．これより，プロセッサ p において最適な候補 $m_{opt}(p)$ が定まる．

$$m_{opt}(p) = \arg\min_{m}\{t_{pm}\}$$

$M+1$ 回目の反復から K 回目の反復までは，各プロセッサで最適候補を採用する．すなわち，これらの反復の実行時間は $\max\{t_{pm_{opt}(p)}\}$ である．

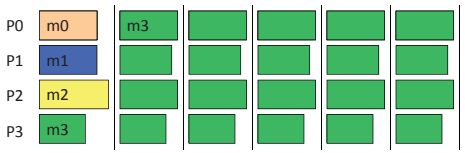

図 3 PEO
Fig. 3 PEO

第 2 の戦略は **PEO** (Parallel Experiments Once) である（**図 3**）．この戦略では，最初の $\lceil M/P \rceil$ 回の反復において，M 個の候補を P 個のプロセッサに分配し，並列に性能評価をさせる．M が P の整数倍でない場合には，複数のプロセッサで性能評価が行われる候補も存在する．このような場合も含めて，探索における候補 m の平均実行時間を t_m とし

よう．$\lceil M/P \rceil + 1$ 回目から K 回目までの反復においては，t_m を最小とする候補

$$\tilde{m}_{opt} = \arg\min_{m}\{t_m\}$$

をすべてのプロセッサで採用する．

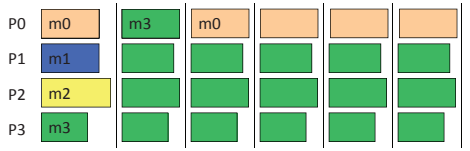

図 4 MPEO
Fig. 4 MPEO

第 3 の戦略は **MPEO** (Modified PEO) である（**図 4**）．この戦略では，最初の $\lceil M/P \rceil$ 回の反復では，PEO と同様に M 個の候補を 1 回ずつ以上評価し，$\tilde{m}_{opt}$ を決定する．$\lceil M/P \rceil + 1$ 回目の反復では，すべてのプロセッサで $\tilde{m}_{opt}$ を使用する．残りの反復，すなわち $\lceil M/P \rceil + 2$ 回目から K 回目までの反復では，各プロセッサが，それまでに自分が性能評価した候補の中で最も所要時間の短かった候補，つまり

$$\tilde{m}_{opt}(p) = \arg \min_{k=1}^{\lceil M/P \rceil + 1} t_{pm(p,k)}$$

を採用する．

$$t_{p\tilde{m}_{opt}(p)} \leq t_{p\tilde{m}_{opt}}$$

となるので，MPEO の合計実行時間は PEO の合計実行時間以下となる．

3. 並列プログラムの自動チューニングの 3 つの問題設定

前の発表[6] では，著者は「並列プログラムの自動チューニング」というひとつの問題に対して 3 つの解を与えるとして SEO, PEO, MPEO を提案した．

しかし，前節の記述では，異なるプロセッサ p, q における同一候補 m の所要時間 t_{mp} と t_{mq} の関連について何も仮定されていない．本稿ではこの点についてより仮定を明らかにする．そして，以下のような 3 通りの問題設定を提案する．

(1) （**無関連タスク問題**）異なるプロセッサ p, q に対して，候補 m の所要時間 t_{mp} と t_{mq} の間に何の関連もない．各プロセッサは，他のプロセッサにおける所要時間情報

を一切用いることなく，自分が使う候補を選択しなければならない．

(2) （**同一タスク問題**）所要時間は，処理するプロセッサによらない．すなわち，候補 m の共通の所要時間 t_m があって，

$$t_{mp} = t_m \qquad p = 1, 2, \ldots, P$$

である．あるプロセッサ p において候補 m が 1 度でも実行されて所要時間 t_m が得られれば，候補 m の性能は完全に既知である．

(3) （**相関タスク問題**）異なるプロセッサ p, q に対して，候補 m の所要時間 t_{mp} と t_{mq} は同一とは限らないが，ある相関がある．(相関を数学的にどう定式化するかは複数の可能性がある.)

ここで「タスク」と呼んでいるのは，各プロセッサにおける計算である．これらの 3 つの問題設定「無関連タスク問題」,「同一タスク問題」,「相関タスク問題」は，前節で導入した SEO, PEO, MPEO におよそ対応している．

無関連タスク問題は，タスクの所要時間に相関がないということとは異なる．タスクの所要時間に関する仮定というより，むしろ解（実験計画）に対する制約である．同期以外のプロセッサ間の通信が困難な場合などが対応すると考えられるが，やや現実味の低い問題設定である．同一タスク問題は，計算を均一に分割してプロセッサに割り当てた場合に相当すると思われる．しかし，所要時間にばらつき（擾乱）があると，あるプロセッサで所要時間を 1 度測定しただけでは，その候補の性能が完全にわかったとは言えないため，厳密に同一タスク問題と同定できる問題が現実にあるかどうかは疑問である．しかし，同一タスク問題に近い問題というのは存在するであろう．相関タスク問題は，現実の並列処理においてより広く見られる状況と考えられる．例えば論文 6) では，疎行列の格納形式に関してオンラインチューニングを行う例題を挙げているが，これは（著者の試した範囲ではすべての場合に）相関タスク問題に相当すると思われる．

4. Bayes 統計を用いた並列プログラムの自動チューニング手法

本節では，前節で導入した無関連タスク問題，同一タスク問題の 2 つに対する Bayes 統計を用いた実験計画法を提案する．相関タスク問題については，紙数の都合で収めきれなかったので，別の機会に報告する．

4.1 無関連タスク問題

無関連タスク問題では，各プロセッサは他のプロセッサで得られた性能情報を参照することなくチューニングを行わなければならない．各プロセッサが独立に Bayes 的オンライン自動チューニングを行えば，無関連タスク問題に対するひとつの Bayes 的手法となる．著者は逐次プログラムのための Bayes 統計に基づく手法を提案してきた[2] ので，これを用いることができる．

しかし，各反復の所要時間は各プロセッサでの所要時間の最大値になることを考えると，プロセッサが連携して，所要時間の期待値が大きめの候補を選択する「探索的選択」をするか，所要時間の期待値が小さい候補を選択する「活用的選択」をするか，歩調を合わせることで，性能を改善できる可能性もある．たとえば著者[8),9)] が提案している「無限希釈」における希釈関数を各プロセッサで共通にすることにより，これが実現できる．

4.2 同一タスク問題

次に，同一タスク問題について考える．まず，同一タスク問題に対して Bayes モデルを導入する．Bayes モデルでは，候補 m の所要時間 t_m はある確率密度関数（事前確率）$p_m(t_m)$ に従って発生すると考える．事前確率は経験 Bayes 法や階層 Bayes 法により実験的に構成することも多いのだが，ここではあらかじめ与えられているとする．特に本稿では事前分布として正規分布を仮定し

$$t_m \sim N(\mu_m, \tau_m^2)$$

とする．以下では，事前分布のパラメタ μ_m, τ_m^2 が候補 m により異なるか否かによっていくつかの場合にわけて論じる．

4.2.1 期待値・分散共通の場合の第 1 反復

まず，すべての候補 m に関して μ_m も τ_m^2 も等しい

$$\mu_m = \mu, \quad \tau_m^2 = \tau^2 \qquad m = 1, 2, \ldots, M$$

と仮定する．そして，これ以外の情報がない第 1 反復での実験計画について考える．

第 1 反復で使用される候補の数を n とする．各プロセッサがひとつずつどれかの候補を選択するので，$1 \le n \le \min\{P, M\}$ である．事前情報において期待値にも分散にも違いがないので，候補は 1 から n までが選ばれると仮定して一般性を失わない．候補 m の所要時間はプロセッサによらず t_m である．よって第 1 反復の所要時間は

$$W_n^1 = \max_{m=1}^{n}\{t_m\}$$

である．

第 2 反復から第 K 反復までの所要時間を以下のように見積もる．すなわち，第 1 反復で性能が測定された n 個の候補のうち，所要時間が最小だったものを，残りの $K-1$ 反復で使用すると考える．すなわち第 2 回の反復の所要時間の推定値は

$$W_n^2 = \min_{m=1}^{n}\{t_m\}$$

となる．よって，第 1 回から第 K 回までのすべての反復の所要時間の期待値は

$$W_n = \max_{m=1}^{n}\{t_m\} + (K-1)\min_{m=1}^{n}\{t_m\} \quad (1)$$

なお，後述のように実際には第 2 反復以降も実験をする可能性はあるのだが，第 1 反復の実験を計画する段階ではそれを無視するという近似を上記の評価では行っている．これを**ワンステップ近似**と呼ぶ．この考え方は逐次計算のためのオンライン自動チューニング[2)]およびオフライン自動チューニング[1)]で著者が用いてきたものである．

さて，確率的に独立な確率変数 x_i $(i = 1, 2, \ldots, n)$ があって，その累積密度関数を $F_i(x_i)$ とすると，$y = \max_{i=1}^{n}\{x_i\}$ の累積密度関数は

$$G(y) = \prod_{i=1}^{n} F_i(y)$$

と表される．これを 1 回微分することで y の確率密度関数 $g(y)$ が

$$g(y) = \sum_{i=1}^{n} f_i(y) \prod_{j=1, j\neq i}^{n} F_j(y)$$

のように得られる（ただし $f_i(x) = dF_i(x)/dx$）．よって y の期待値 $E(y)$ は

$$E(y) = \int_{-\infty}^{\infty} y g(y) dy$$

という 1 次元積分で得られる．もし確率密度関数がすべて同じであれば

$$g(y) = nF^{n-1}(y)f(y)$$

であり，容易に計算できる．

これを用いて正規分布 $N(\mu, \tau^2)$ に独立に従う確率変数 t_m の最大値および最小値の分布を評価すると，

$$\max_{m=1}^{n} = \mu + S_n\tau, \qquad \min_{m=1}^{n} = \mu - S_n\tau$$

のようになる．ここで S_n は μ, τ によらない定数であり，添え字 n についての関数とみると単調増加である．**図 5** は S_n を $\log_2 n$ に関してプロットしたものである．計算には Scilab を用いた．

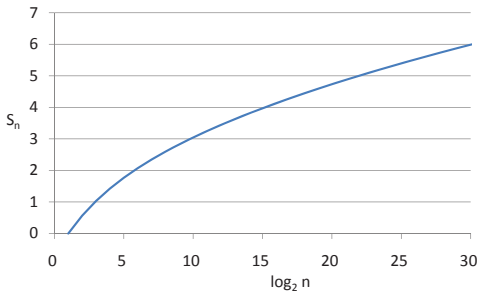

図 5 S_n
Fig. 5 S_n

以上の結果を式 (1) に代入すると

$$W_n = K\mu + (2-K)S_n\tau$$

となる．ここで反復回数 K を 3 以上とすると，$2 - K < 0$ である．S_n は n に関して増加関数であるから，n が大きいほど所要時間の期待値 W_n は小さい．よって n を取りうる最大値に選ぶのが最適である．これは

$$n = \min\{P, M\}$$

ということになる．すなわち，**第 1 反復では，すべてのプロセッサができるだけ異なる候補を選択して性能評価をする「並列実験」が最適**である．

この結論は反復回数が $K > 2$ で事前分布が対称であれば成立する．プロセッサ数 P にも，事前分布のパラメタ μ および τ にもよらない．よって，事前分布を具体的に指定することなく，第 1 反復では並列実験を行ってよい．その後，$M > P$ であれば，P 個の候補についての所要時間 $(t_m)_{m=1}^{P}$ が得られるので，これを参照し，経験 Bayes ないし階層 Bayes の手法によって事前分布を構築することができる．ここから以下のように第 2 反復を考えてよい．

4.2.2 期待値・分散共通の場合の第 2 反復以降

次に第 2 反復について考える．第 3 反復以降も同様にして評価することができる．

以下では，第 1 反復で実行された候補のうち，所要時間が最短だったものを m_{opt}，その所要時間を t_{opt} とする．なお，$M \leq P$ の場合には m_{opt} が最適候補であり，最適解が自明である．このため以下では $M > P$ を仮定する．

前述のように，第 1 反復におけるワンステップ近似に関わらず，第 2 反復においては探

索をする可能性を改めて考える．すなわち，次の 2 つの場合を考察する．(i) 第 2 反復以降ではすべてのプロセッサがすべての反復で m_{opt} を採用する．このとき第 2 回から第 K 回までの反復の所要時間の合計は

$$W_0 = (K-1)t_{opt} \tag{2}$$

である．(ii) 第 2 反復では，第 1 反復で選択されなかった候補 $n \leq P$ 個（$P+1$ から $P+n$ までと仮定してよい）を選択して実行する．第 3 反復からあとは，第 1 反復と第 2 反復で実行された合計 $P+n$ 個の候補のうち所要時間が最短のものを選択する．このとき第 2 回から第 K 回までの反復の所要時間の合計の期待値は

$$W_n = E\left(\max_{m=P+1}^{P+n}\{t_m\}\right) + (K-2)E\left(\min\{t_{opt}, t_{P+1}, t_{P+2}, \ldots, t_{P+n}\}\right) \tag{3}$$

となる．

これより，W_0 および W_n を $n = 1, \ldots, \min\{P, M-P\}$ について計算し，W_n が最小となる n を求めればよい．この W_n は

$$W_n = (K-1)\mu + S_n\tau - (K-2)S_n((\mu - t_{opt})/\tau)\tau$$

という形になる．ここで $S_n(t)$ は n に関して増加，t に関して減少関数で，

$$S_0(t) = 0, \qquad \lim_{n\to\infty} S_n(t) = \infty, \qquad \lim_{t\to-\infty} S_n(t) = S_n, \qquad \lim_{t\to\infty} S_n(t) = 0$$

である．

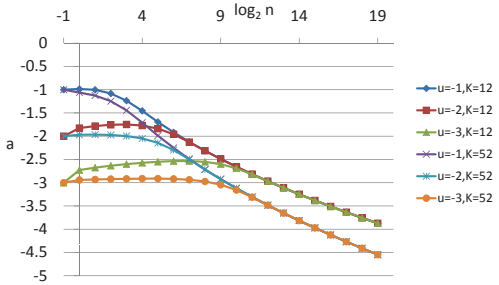

図 6 $W_n = (K-1)(\mu + a\tau)$ の a を $\log n$ に対してプロット．ただし W_0 は -1 に表示，$t_{opt} = \mu + u\tau$
Fig. 6 Plot of a where $W_n = (K-1)(\mu + a\tau)$, W_0 plotted at -1. $t_{opt} = \mu + u\tau$

実際に計算した例を**図 6** に示す．次のことが観測される．(a) K が小さく t_{opt} も小さいときは W_n は n がある程度小さい範囲では n に関して増加関数である．しかし n がある程度大きいところでは n に関して減少に転じる．(b) K が大きく t_{opt} も大きいとき，W_n は n に関して減少関数である．これらの例では，$W_0 > W_P$ であれば，$n = P$，すなわち並列実験が最適である．そうでなければ実験をやらず，第 1 反復で見つかった m_{opt} を採用するのが最適である．

いまのところ $W_0 > W_1 < W_2$ となるような例は見つかっていないが，非存在を証明できていない．もしそのような例があれば，$P = 2$ のとき $n = 1$ が最適ということになる．従って，上記 (a), (b) の観測が一般的に正しければ，W_0, W_1, $W_{\min\{P,M-P\}}$ **の 3 つを計算し，最小のものに対応する実行を行うのがよい**と考えられる．

本手法の特徴のひとつは，候補の数 M が不足する場合を除いて，実験計画が M に依存しないことである．このため，候補数 M が有限である必要はなく，無限個の候補があっても本手法は適用できる．

4.2.3 シミュレーションによる評価

ここでは，従来の網羅的実験を行う PEO と，本稿で提案した Bayes 的並列実験手法とをシミュレーションで比較する．

表 1 Bayes 手法の評価：全実行時間 $K\mu + a\tau$ の a 部分
Table 1 Evaluation of Bayesian method: a of total execution time $K\mu + a\tau$

M	P	K	PEO	Bayes	平均実験回数
100	10	20	−10.4	−28.4	2.4
100	10	50	−83.8	−87.3	6.1
100	10	200	−467	−473	7.3
100	10	1000	−2.66e+3	−2.67e+3	8.3
1000	100	20	−8.00	−39.36	1.8
1000	100	50	−107	−123	3.0
1000	100	200	−574	−581	6.1
1000	100	1000	−3.17e+3	−3.14e+3	9.0
1000	25	50	48.4	-105	3.7
1000	25	200	-412	-501	15.1
1000	25	1000	−2.94e+3	−2.90e+3	32.8

表 1 に評価結果を示す．M, P, K をいろいろ変えて，10 回のシミュレーションを行った平均を示す．数値は全実行時間が $K\mu + a\tau$ となるところの a である．また，Bayes 手法に関しては，実験が行われた平均回数も示した．有効数字 3 桁で表示してある．

M/P に比べて K があまり大きくはない場合には，多数の実験を行うよりも，適度に少

ない実験結果で留めておくのがよい．このような場合には，Bayes 手法は PEO よりも明確に短い全実行時間を達成する．M/P に比べて K が大きい場合には，多めに実験をしてよいものを探索するのがよい．そのような場合であっても，提案手法はよい結果が得られれば実験を停止し，よい結果が得られなければ実験を継続するという性質があるため，平均実験回数は M/P よりも小さめになる．それでも多くの場合において PEO よりも少ない全実行時間となっている．いくつかの例において Bayes が PEO に劣っている結果が出ているが，わずか 10 回の実験でもあり，確率的にはこのような事例も存在しなければならない．なお，$M = 1000$, $P = 25$, $K = 50$ の例では PEO が正の a を出している．これは $M/P = 40$ が K に近すぎる例で，この場合は実験を一切やらずに任意の一つの候補を選択した方がよい．

4.2.4 その他の場合

最後に，モデル等により事前情報を持っていて，第 1 反復の前に候補ごとに事前分布の期待値が異なる場合について考える．

まず，分散が共通な場合を考える．この場合，事前情報において速いと予測された候補から順番に試してみるのが最適である．そこで，候補は事前分布の期待値が小さいものから順番にソートされているとする．

すると，評価すべき式としては式 (1) と同一になる．期待値の計算は 1 次元積分であるが，事前期待値が同一の場合のようにひとつのパラメタ S_n では表現できず，個別の事例に対して数値計算する必要がある．$W_1, W_2, \ldots, W_{\min\{P,M\}}$ を評価し，最小となる W_n を選択する．W_n が n に関して単調減少等とは言えない．第 2 反復以降の式も，(2) および (3) により評価をすればよい．やはり評価値 W_n を最小にする n を決定する．

最後に，事前分布が分散も期待値もばらばらな場合を考える．このような場合には，事前情報において遅いと予測されたものをあえて試した方がよい可能性がある．すなわち，「いくつ選ぶか」ではなく，「どの組み合わせを選ぶか」という問題となる．このため最適な実験計画について網羅的に探索するには，プロセッサ数に関して指数関数的な時間がかかる．これは現実的ではないので，発見的近似解法が必要と考えられる．

5. おわりに

本稿では，並列プログラムの局所的チューニングパラメタのオンライン自動チューニングのための Bayes 的手法について論じた．まず，無関連タスク問題，同一タスク問題，相関タスク問題という 3 つの問題クラスを導入した．その後，無関連タスク問題と同一タスク問題に対して Bayes 的手法を示した．同一タスク問題で，事前情報において全候補の期待値・分散が共通の場合について詳述し，シミュレーションによって従来法である PEO と比較をした．

今回詳しく論じたのは全候補の期待値・分散が共通の場合だけである．モデル等により事前に有望な候補などがわかっている場合については 4.2.4 節で若干論じたが，より詳細な検討が必要である．特に，期待値も分散も異なる場合については，具体的なヒューリスティクスを示し，評価をする必要がある．このほか，今回は分散が既知という仮定になっている．実際には分散が既知ということは少ないため，実験データから分散を推定しながら実験計画を行うのが現実的である．

今回は紙数の都合により相関タスク問題について論ずることができなかったので，別の機会に報告する．相関タスク問題は同一タスク問題よりも格段に難しいため，近似的な理論やヒューリスティクスによる手法が必要である．

また，5) で論じた並列試行についても Bayes 的手法が必要である．並列試行と並列実験の融合についても，Bayes 的手法の中で考えてゆきたい．

謝辞 有益な議論をいつもいただいています自動チューニング研究会（`http://atrg.jp`）のみなさまに感謝いたします．

本研究の一部は JST 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」，科学研究費「メニーコア・超並列時代に向けた自動チューニング記述言語の方式開発」の支援を受けています．

参考文献

1) 須田礼仁：オフライン自動チューニングの数理手法，情報処理学会研究報告 HPC-125-3, pp.1–9 (2010).
2) Suda, R.: A Bayesian Method for Online Code Selection: Toward Efficient and Robust Methods of Automatic Tuning, *Proc. iWAPT 2007*, pp.23–31 (2007).
3) 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 HPC-116，pp.109–114 (2008).
4) 須田礼仁：並列計算機におけるソフトウェア自動チューニングのための数理モデル，日本応用数理学会 2009 年度年会，pp.13–14 (2009).
5) 須田礼仁：並列試行による並列処理のためのオンライン自動チューニング，計算工学講演会論文集，Vol.15, No.1, pp.89–92 (2010).
6) Suda, R.: Methods of Parallel Experimental Design of Online Automatic Tuning and their Application to Parallel Sparse Matrix Data Structure, *Proc. iWAPT 2010*

Vol.2010-HPC-126 No.41
2010/8/5

(in Proc. VECPAR'10) (2010).
7) Suda, R.: Online Automatic Tuning of Parallel Sparse Matrix Computations, *PMAA 2010 (oral presentation)* (2010).
8) Suda, R.: *A Bayesian Method of Online Automatic Tuning*, Software Automatic Tuning: Concepts and State-of-the-Art Results, chapter 16, Springer (2010). to appear.
9) 須田礼仁：自動チューニングのための数理基盤技術，応用数理 (2010). (予定).