Vol.2010-HPC-125 No.3
2010/6/17

# オフライン自動チューニングの数理手法

須　田　礼　仁[†1]

本稿ではオフライン自動チューニングの数理手法について論ずる．まず，自動チューニングの基礎概念についてオフライン自動チューニングを想定して再検討をする．具体的には，オフライン自動チューニングの一般論として，チューニングに関わる実行条件（静的・動的，観測可否，制御可否，特徴量）を整理し，それに対する適応方法（定数的適応，関数的適応）を考察する．次に，チューニングパラメタ制御関数とコスト推定関数を定義し，これらを用いた関数的適応の 6 つの実現方法を論じる．また，最適化問題としての定式化の方法を例示する．この定式化に基づき，オンライン自動チューニングよりもオフライン自動チューニングが望ましい状況について考察する．次に，目的関数のひとつである実施時コストの定式化について論じ，Bayes 統計を用いた推定法を考える．このとき，特徴量を用いない基本的な場合については，試行コストと実施コストの重みつき和で目的関数を定式化すると，オフライン自動チューニングは「逐次サンプリング問題」に帰着される．その数学的な最適解（既知）を紹介し，実用的な近似解であるワンステップ近似を提案する．シミュレーションにより，ワンステップ近似を評価したところ，試行コストと実施コストがほぼ均衡する実験計画が得られた．

# Mathematical Methods of Offline Automatic Tuning

Reiji Suda[†1]

We discuss mathematical methods of offline automatic tuning. First, concepts of automatic tuning are reformulated for application to offline automatic tuning. As general discussion of offline automatic tuning, the execution conditions related tuning are classified into static or dynamic conditions, observable or non-observable conditions, controllable and non-controllable conditions, and features and non-features. Constant adaptability and functional adaptability are introduced. Next, six methods of functional adaptability are discussed by introducing tuning parameter control function and cost estimate function. Some methods of formulation of offline automatic tuning problem as optimization problem are exemplified. Based on those formulations, we discuss possible situations where offline automatic tuning is preferred than online automatic tuning. Next, formulations of practical execution costs are discussed, and estimation methods based on Bayesian statistics are introduced. Optimal offline automatic tuning is then formulated as a sequential sampling problem, if the objective function is a weighted sum of trial execution costs and practical execution costs, and no features are used. The mathematical optimal solution (known) is explained, and a practical approximate solution, called one-step approximation, is proposed. The one-step approximation is evaluated through simulations, where the trial execution costs and the practical execution costs are balanced.

## 1. はじめに

自動チューニングは，ソフトウェアの実装に自由度（チューニングパラメタ）を残し，実環境でソフトウェアを実行して性能を測定することにより，チューニングパラメタを調節し，多様な環境で高い性能を実現することを目指すパラダイムである．マルチコア CPU の普及，深いキャッシュ階層，GPU の汎用計算への利用の拡大，スーパーコンピュータにおけるプロセッサ数の爆発的増大など，ハイパフォーマンスコンピューティングにおけるハードウェア条件は多様性を増している．このため各プラットフォームに対して手作業でチューニングを行うのは困難となっており，自動チューニングが必須の技術として認識されるに至っている．

我々は自動チューニングの数理手法について研究を行い，主にこれまでオンライン自動チューニングの数理手法について論じてきた[1)]．これに対して，本稿ではオフライン自動チューニングとその数理手法について論ずる．

## 2. 自動チューニングの概念

まず本節では，自動チューニングの基本的な概念について再考する．これまで著者は何度かオンライン自動チューニングを想定して自動チューニングの諸概念について論じてきたが，オフライン自動チューニングを導入するに当たっては，いくつかの点において再考が必要となる．

**チューニングパラメタ**は，ソフトウェア中に埋め込まれた自由度である．自由度にはさまざまなものがありうるが，(1) ループ変換のように，演算内容もデータ構造も変化させない

†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

**スケジューリング変種**，(2) データ構造やメモリ上のデータ配置を変化させる**データ構造変種**，(3) 同様の結果を出す異なるアルゴリズムを選択する**アルゴリズム変種**, (4) SIMD 命令，GPU 向けコーディング，明示的並列化などの**プラットフォーム特有コーディング**の 4 種類が主なものと考えられる．自動チューニングの対象は単一のソフトウェア，複数のソフトウェアの組み合わせ，ハードウェア等も含む場合などが考えられるが，ここでは簡単のため単一のソフトウェアを仮定し，それを**ターゲットソフトウェア**と呼ぶ．

自動チューニングは，実環境で性能評価をしながらチューニングパラメタを調整し，最適な性能を達成することを目指す．我々は，**性能**の最大化を**コスト**の最小化ととらえる．コストには，計算時間，消費電力，課金，計算精度などが考えられるが，本稿では別途指定しない場合，計算時間を考える．

性能評価の方法には 2 つの種類がある．ひとつめは**実施**で，これは，ターゲットソフトウェアの実用的な使用機会である．ふたつめは**試行**で，これは，性能評価のためだけのターゲットソフトウェアの実行であり，計算結果は利用されない．我々は実施のみを用いる自動チューニングを**オンライン**, 試行のみで自動チューニングする場合を**オフライン**と呼んでいる．当然，実施と試行の両方を利用する**オフライン・オンライン混合自動チューニング**も考えることができる．

自動チューニングは実際の実行条件下で性能測定をしてチューニングを行うのが特徴である．オンライン自動チューニングでは，実施のみを用いるので，実際の実行条件下でチューニングが行われる．このため,「実際の実行条件」の具体的な中身について詳しく議論をする必要がなかった．しかし，オフライン自動チューニングでは試行のみを用いるため，実行条件は人工的に作らなければならない．このため,「人工的に与えた実行条件がターゲットソフトウェアの実用的な実行における条件を適切に近似しているかどうか」という問題を吟味する必要が生じる．そこで，以下ではソフトウェアの実行条件について考える．

ターゲットソフトウェアの実行条件には，いくつかの要因が考えられる．(1) **ハードウェア条件**は，処理が行われるハードウェアである．CPU アーキテクチャやキャッシュサイズ，ネットワークインタフェースやネットワークトポロジーなどを含む．(2) **ソフトウェア条件**は，ターゲットソフトウェアを取り巻くソフトウェアである．たとえばアプリケーションソフトウェアから見ると，数値ライブラリや通信ライブラリ，コンパイラなどがソフトウェア条件となりうる．(3) **データ条件**は，処理対象のデータに関する情報である．疎行列計算ライブラリであれば，行列サイズ，疎行列構造，対称性，優対角性，固有値分布などが考えられる．(4) **環境条件**は，自動チューニングの対象となるソフトウェア以外のプロセスや，システムを共用している他のユーザで，大域ネットワークの性能を考えるとき，常に無視できない．

上記のように，ソフトウェアの実行条件にはさまざまな要因がある．これらの条件のなかには，ターゲットソフトウェアの性能に影響を与える条件もあり,（あまり）与えない条件もある．また，**観測できる条件**と**観測できない条件**が考えられる．ここで，観測は実行（試行あるいは実施）の前（直前）に行うものとする．観測にはコストがかかる場合が考えられるので，観測をしないほうが効率的な場合がある．たとえば疎行列の固有値は反復法の性能に極めて重要な影響を与えるデータ条件で，CG 法のアルゴリズム中に現れるパラメタから近似的に観測可能であるが，正確に同定するのは計算量的に有利ではない．このため，自動チューニングにおいては観測できる条件をすべて利用するとは限らず，むしろその一部を実行条件の**特徴量**として選択する．観測できる条件のうち何を特徴量として選択するか（特徴量選択）は自動チューニングの数理的課題の一部であり，自動・半自動・手動のアプローチが考えられる．ただし本稿ではこれはこれ以上論じない．

さらに，試行において人工的に与えられる条件に関して，**制御できる条件**と**制御できない条件**とが考えられる．ここでも，両者の区別は明確ではない．たとえば，直接的には制御できないが，間接的な手段で条件をある程度変更させることができる場合も考えられる．さて，試行において制御できる条件を適切に制御することは基本的に重要である．他方，制御できない条件の試行における扱いは，まず**静的な条件**と**動的な条件**に分けて考える．たとえば，他のプロセスがターゲットソフトウェアに与える影響は動的ととらえることもできる．しかし，その変動のしかたが一定であると仮定すれば，適当な時間だけ平均して考えると，平均部分は静的であり，変動部分のみが動的と考えることができる．静的な条件は，明示的に制御できなくても，試行時と実施時において同じとなることに注意する．次に，制御できない動的な条件は，観測できるか否かで扱いが分かれる．観測できる条件は，特徴量として採用することで，自動チューニングのモデルに含めることが可能である．最後に，制御も観測もできない動的な条件は，試行時と実施時で条件に差を生じることになる．制御できる条件についても，特徴量として採用しない場合，および実施時の条件を試行時に十分な精度で再現しない場合には，変動分が誤差となる．

上記の議論を明確にするために，2 つの例について論ずる．(1) 試行時には制御できるが，実施時には制御できない条件．たとえば，試行時には仮のデータを与えて実行させるのでデータ条件は制御できるが，実施時には応用で解かなければならない問題が与えられるのでデータ条件は制御できない．制御できるかどうかの分類は試行時の制御可能性で決める．す

なわち，実施時に制御できない条件も「制御できる」条件に分類されうる．(2) 試行時には観測（または制御）できるが，実施時には観測できない条件．観測できないと特徴量にはならないので，必然的に非特徴量になる．たとえば，事後にしか観測できない条件は，チューニングパラメタ制御関数の特徴量とすることができない．観測できるかどうかの分類は実施時の観測可能性で決める．すなわち，実施時に観測できない条件は「観測できない」条件に分類する．

**表 1** オフライン自動チューニングにおける実行条件の分類と適応方法
Table 1 Classification of execution conditions and adaptability in automatic tuning

| 条件の種類 | | 適応方法 |
|---|---|---|
| 静的な条件および動的な条件の平均部分 | | 定数的適応 |
| 動的な条件の変動部分 | 特徴量（制御または観測される） | 関数的適応 |
| | 非特徴量（制御も観測もされない） | 非適応 |

以上の視点で実行条件を分類すると**表 1** のようになる．表 1 には，オフライン自動チューニングにおける条件への適応方法についても付記してある．**定数的適応**とは，試行における性能評価と，それに基づくチューニングにより，暗黙のうちに実現される適応性である．また，**関数的適応**とは，特徴量を参照してチューニングパラメタが設定されることにより，明示的に実現される適応性である．定数的適応は，関数的適応の定数部分として実現することが可能である．

関数的適応の実現方法にはいくつか考えられる．これらを整理して導入するため，まず 2 種類の関数を定義する．ひとつは，特徴量を引き数として，チューニングパラメタを出力する関数で，これを**チューニングパラメタ制御関数**と呼ぶことにする．すなわち，特徴量を $x$, チューニングパラメタを $t$ とし，チューニングパラメタ制御関数を $\tilde{t}(x)$ としたとき，$\tilde{t}(x) \approx t_{opt}(x)$ である．ここで，コスト関数を $c(t,x)$ としたとき，$c(t_{opt}(x),x) = \min_t c(t,x)$ である．もうひとつは，特徴量とチューニングパラメタを引数として，推定コストを出力する関数で，これを**コスト推定関数**と呼ぶことにする．すなわち，コスト推定関数を $\tilde{c}(t,x)$ とすると，これは $\tilde{c}(t,x) \approx c(t,x)$ である．

さて，チューニングパラメタ制御関数を使うか否か，コスト推定関数を使うか否か，また，それぞれ使う場合には，ソフトウェア開発者の事前知識に基づいて関数を定義するか，試行の情報を用いて関数を定義するか，という視点から，関数的適応が 6 通り考えられる．

(1) **コスト推定関数なし，事前知識によるチューニングパラメタ制御関数（C0P1）**．ソフトウェア開発者により，チューニングパラメタ制御関数が組み込まれる．

(2) **コスト推定関数なし，試行実験によるチューニングパラメタ制御関数（C0P2）**．ソフトウェア開発者がチューニングパラメタ制御の関数族を与え，その中から試行実験により優れた性能を与えるものを選択する．

(3) **事前知識によるコスト推定関数，チューニングパラメタ制御関数なし（C1P0）**．ソフトウェア開発者がコスト推定関数を与え，そのコストを最小にするようにチューニングパラメタが制御される．

(4) **事前知識によるコスト推定関数，試行実験によるチューニングパラメタ制御関数（C1P2）**．ソフトウェア開発者は，チューニングパラメタ制御の関数族と，コスト推定関数とを与える．コスト推定関数を用いてチューニングパラメタ制御関数を評価し，優れた性能を与える関数を選択する．

(5) **試行実験によるコスト推定関数，チューニングパラメタ制御関数なし（C2P0）**．ソフトウェア開発者は，コスト推定の関数族を与え，その中から試行実験によりコストをよく推定する関数を選択する．実行時には推定されたコストを最小にするようにチューニングパラメタが制御される．

(6) **試行実験によるコスト推定関数，試行実験によるチューニングパラメタ制御関数（C2P2）**．ソフトウェア開発者は，チューニングパラメタ制御の関数族とコスト推定の関数族を与える．試行実験によりコストをよく推定する関数を選択する．さらに，選ばれたコスト推定関数を用いてチューニングパラメタ制御関数を評価し，優れた性能を与える関数を選択する．

これらのうち，C0P2, C2P0, C2P2 の 3 つは試行実験に基づいて関数的適応を実現している．これに対し，C0P1, C1P0, C1P2 の 3 つは事前知識に基づいて関数的適応を実現している．後者であっても，定数的適応を試行実験に基づいて行うことにより，自動チューニングが実現できる．また，オフライン自動チューニングでは，試行においてのみコストの測定が行われ，実施時には行われないと仮定する．このため，チューニングパラメタ制御関数もコスト推定関数も，試行が完了した時点で固定される．実施時には，そのときの特徴量を参照して，チューニングパラメタが決定される．なお，オンライン自動チューニングにおいても上記とほぼ同じ 6 通りが考えられる（試行ではなく，実施により性能評価を行う）．

自動チューニングは一種の最適化であるから，目的関数と制約条件を明確にすべきである．当然のことながら，チューニングは未来の実施におけるコストの低減のために行われるのであるから，目的関数は未来の実施におけるコストを中心に定義されなければならない．

従って，未来の実施における実行条件を推定することが必要不可欠になるが，これには**計画ベースの推定**と**履歴ベースの推定**がある．計画ベースの推定では，ターゲットソフトウェアがどのような実行条件でどれだけ使用されるか，あらかじめ計画を立てておく．履歴ベースの推定では，過去の使用履歴から将来どのように利用されるかを推定する．

自動チューニングにおいて，試行のコストは無視できない．試行コストの扱いを一般的な最適化の枠組みで定式化する方法はいくつか考えられる．第 1 の方法では，未来の実施コストと試行コストの和を目的関数とする．これを**コスト総和による定式化**と呼ぶことにする．第 2 の方法では，試行コストを一定値以下にするように制約し，実施コストを最小化する．これを**試行コスト制約による定式化**と呼ぶ．第 3 の方法では，実施コストが一定値以下になるまでチューニングをするという制約を与えて，試行コストを最小化する．これを**実施コスト制約による定式化**と呼ぶ．これらの定式化を用いると，自動チューニングの問題が最適化の問題として定式化できる．しかし，現実に開発者やユーザが期待しているような最適化が実現されるかどうかという問題については慎重な議論が必要である．試行コストと実施コストという 2 つの目的関数を持つ問題として，様々な数理的定式化を検討する余地がある．この点に関しては，オンライン自動チューニングでは問題が簡単である．すなわち，試行がなく実施のみであるため，実施コストのみの単一目的関数の最適化問題として自然に定式化される．

オフライン自動チューニングにおける試行は，性能評価にしか用いられず，その計算結果は捨てられる．この特性を利用すると，意図的に実施時とは異なる計算を実行させることにより試行コストを低減する工夫が可能になる．第 1 に挙げるのは**中断**である．たとえば試行コストとして所要時間を考えよう．ひとつの試行を開始してから時間 $\tau$ だけ経過しても計算が終了していないことを確認し，中断する．すると，この試行の所要時間は $\tau$ より大であるという情報が得られる．さらに，擾乱が十分小さく，$\tau$ 以下の所要時間で同じ計算を実現するチューニングパラメタがあることがわかっているなどの仮定がそろえば，$\tau$ を超えて試行計算を継続することに益がないと決定できるであろう．試行コストを低減する第 2 の方法は**問題サイズの縮小**である．すなわち，実際に処理するよりも小さいサイズの問題で性能評価をして，大きなサイズの問題での性能に関する間接的な情報を得る．自動チューニングで対象とするコストに，問題サイズに直接依存する**示量性コスト**と，問題サイズに依存しない（問題の性質に依存する）**示強性コスト**を想定しよう．示強性コストは，問題の性質を維持したまま問題サイズを小さくすることにより，小さいサイズの問題を処理した際のコストから大きいサイズの問題を処理した際のコストを推定することができる．示量性コストも，問題サイズとコストとの主たる関数性を推定することができると仮定すれば，小さいサイズの問題から大きいサイズの問題の性能情報を抽出することができる．いずれにせよ，小さい問題を解いたときのコストと，大きな問題を解いたときのコストとの間に，何らかの相関が予想される場合には，問題サイズを縮小しても性能情報が得られる．

## 3. オフライン自動チューニングの必要な状況

オフライン自動チューニングは，性能の測定を試行時にのみ行う．すなわち，オフライン自動チューニングは「実施時には性能を測定しない，チューニングパラメタ制御関数やコスト推定関数の修正をしない」という制約条件が課せられた自動チューニングである．本節では，どのような場合にこの制約条件を課すことが適当なのか，考察する．

状況 1（開発時チューニング）：ソフトウェアの開発者が，何らかの理由により，自動チューニング機構をソフトウェアに組み込みたくないという場合が考えられる．たとえば，プラットフォームのメモリ量が限られているために実行可能コードのファイルサイズを最小限にしたいとか，バグを最小限にするためにソフトウェアの適応性を固定したいとかいう状況が想定できる．このとき，ソフトウェア開発者は，想定されるハードウェア条件，ソフトウェア条件，データ条件，環境条件を与えてオフライン自動チューニングを行い，チューニングパラメタを固定したオブジェクトコード（または変換されたソースコード）を生成する．

状況 2（チューニングパラメタの実施時固定）：ソフトウェアの利用者が，何らかの理由により，チューニングパラメタを固定して実施したいという場合が考えられる．たとえば，ターゲットソフトウェアを部品として用いているソフトウェアの性能評価のために，ターゲットソフトウェアの性能が変動するのを抑制したいとか，デバッグ等の目的で処理内容に高い再現性を要求したいとかいう状況が想定できる．このとき，ソフトウェア利用者は，想定されるハードウェア条件，ソフトウェア条件，データ条件，環境条件を与えてオフライン自動チューニングを行い，チューニングパラメタを固定してターゲットソフトウェアを実行することで，目的を達成することができる．

逆にオンライン自動チューニングは，「試行をしない」という制約条件が課せられた自動チューニングと考えられる．これらの制約条件を緩和すると，オフライン・オンライン混合自動チューニングが得られる．以下では，オフライン・オンライン混合自動チューニングが有利になりうる状況について考察する．

状況 3（実施コストの抑制）：オンライン自動チューニングにおいては，性能情報獲得のために，コストの大きなチューニングパラメタを実施時に選択しなければならない．しかし，

何らかの事情により，それぞれの実施時のコストを抑制したい（平均的な実施コストは最小化しつつ）という場合がある．このとき，コストの大きい選択肢は実施時に用いないという制約を課し，制約条件を超えるコストが期待される選択肢は試行によって性能評価をする．特に，オンライン自動チューニングでは，チューニングパラメタ制御関数やコスト推定関数がある程度の精度で構築されるまでの「初期段階」で大きなコストがかかるので，オフライン自動チューニングの手法を初期段階に適用することが考えられる．

状況 4（アイドル時自動チューニング）：計算機がアイドルの時間帯に試行を行うことにより，性能情報を収集し，オンライン自動チューニングを補強することが考えられる．アイドル時の試行のコストを無限小と考える場合と，エネルギーなども考慮に入れて有限の（しかし実施時よりは割安の）コストがかかるとみなす場合とが考えられる．後者の場合は，必ずしもアイドル時に試行をするのが最適になるとは限らない．

## 4. オフライン自動チューニングの数理

### 4.1 非特徴量と擾乱

制御される条件や観測できる条件も非特徴量になりうる．特徴量でないということは，これらの条件が異なっていても，チューニングパラメタは同じ値が選択されるということである．一定のチューニングパラメタ設定で複数の条件に対応しなければならないので，(コストそのものではなく）コストの統計量を用いて目的関数や制約条件を定めることになる．観測できる条件の場合，試行時の分布と実施時の分布が異なっていても，両方の分布が既知であれば，重み付けサンプリング等により実施時の分布に対応するコストの統計量を推定することが可能であろう．また，制御される条件を適切に制御して実施時のコストの統計量を求めることは，数値積分（モンテカルロ法を含む）の問題である．これらの問題は自動チューニングの数理として検討すべき課題であるが，以下では簡単のため，制御しない条件は試行時の分布と実施時の分布が同じであるとし，制御される条件は実施時の分布に従う独立な標本として与えられる（モンテカルロ法）とする．したがって非特徴量に関するコスト変動は実施時と同一の分布に従い，標準的な統計手法により統計量を推定することができると仮定する．このような非特徴量は**擾乱**とも呼ばれる．

このとき，コストは $c(t,x,y,u)$ とあらわされる．ここで $t$ はチューニングパラメタ，$x$ は制御される特徴量，$y$ は観測される特徴量，$u$ は非特徴量である．制御される特徴量 $x$ の実施時の分布を $p_x(x)$, 観測される特徴量 $y$ の分布を $p_y(y)$，非特徴量 $u$ の分布を $p_u(u)$ とする．コスト推定関数 $\tilde{c}(t,x,y)$ は，チューニングに利用されない非特徴量に関して平均を取った平均コストを推定する．

$$\tilde{c}(t,x,y) \approx \int c(t,x,y,u)p_u(u)du$$

また，チューニングパラメタ制御関数 $\tilde{t}(x,y)$ は，与えられた特徴量 $x$, $y$ に対して，平均コストを最小にするチューニングパラメタ $t$ を近似的に与える．

$$\tilde{t}(x,y) \approx \mathrm{argmin}_t \int c(t,x,y,u)p_u(u)du$$

### 4.2 実施コストの期待値

次に，試行実験に基づく関数的適応（C0P2, C2P0, C2P2）における実施コストの期待値を考察する．ここでは簡単のため，実施時の条件は完全に既知であると仮定する．実施時の条件が既知でない場合には，実施時の条件の事前分布を与え，それに関する統計量で目的関数や制約条件を定めることになる．

第 1 に，チューニングパラメタ制御関数 $\tilde{t}(x,y)$ が定義されており，コスト推定関数は使われていないと仮定する（C0P2）．このとき，実施時にはチューニングパラメタ $t=\tilde{t}(x,y)$ が選択される．従って，実施時の平均コストは

$$C(\tilde{t}) = \iiint c(\tilde{t}(x,y),x,y,u)p_x(x)p_y(y)p_u(u)dxdydu$$

となる．ここで，想定されているチューニングパラメタ制御関数の空間を $T$ とする．すなわち $\tilde{t} \in T$ である．このとき，C0P2 の問題は，

(1)　与えられた $\tilde{t}$ に対して $C(\tilde{t})$ をどう評価するか
(2)　$T$ の中から $C(\tilde{t})$ を最小にする $\tilde{t}$ をどう探すか

という 2 つの問題に分解できる．しかし，コスト関数に関する仮定なしにこれらの問題を解くのは難しいと思われる．(1) では $x$ および $y$ に関する積分が必要であるが，全数列挙できる場合を除くと，関数 $c$ に関する何らかの仮定がないと積分を近似するのは困難である．(2) では最適な $\tilde{t}$ を探したいが，やはり全数列挙できる場合を除くと，関数 $c$ に関するヒントがなければ最適性を議論することが困難である．ただし，$x$, $y$, $t$ の取りうる範囲がそれぞれ有限（少数）の場合には直接評価できるため，解を求めることができる．

時折用いられている定式化として，チューニングパラメタの最適値

$$t_{opt}(x,y) = \mathrm{argmin}_t \left\{ \int c(t,x,y,u)p_u(u)du \right\}$$

を教師信号として関数 $t_{opt}(x,y)$ を学習することにより $\tilde{t}(x,y)$ を構築することがなされて

いる．しかしこの定式化から得られる $\tilde{t}$ に対する実施コスト $C(\tilde{t})$ がはたして $C(t_{opt})$ にどれほど近いかについて知見を得ようとするなら，コスト関数 $c(t,x,y,u)$ に関する仮定（学習においては利用されていない）が必要である．

第 2 に，チューニングパラメタ制御関数が定義されていない場合を考える（C2P0）．この場合，コスト推定関数 $\tilde{c}(t,x,y)$ を用いて

$\tilde{t}_{opt}(x,y) = \mathrm{argmin}_t\{\tilde{c}(t,x,y)\}$

のようにチューニングパラメタが選択される．すなわち，問題は

(1) 試行時にコストを推定する $\tilde{c}(t,x,y)$ をどのように構築するか

(2) 実施時に与えられる $x$, $y$ に対して，コスト推定関数を最小にする $\tilde{t}_{opt}(x,y)$ をどのように求めるか

という 2 つの問題に分割できる．このとき実施時の平均コストは

$$C(\tilde{t}_{opt}) = \iiint c(\tilde{t}_{opt},x,y,u)p_x(x)p_y(y)p_u(u)dxdydu$$

となる．ここで $\tilde{c}(\tilde{t}_{opt},x,y,u) - \tilde{c}(t_{opt},x,y,u) \leq 0$ に注意すると，最適解 $t_{opt}$ と近似解 $\tilde{t}_{opt}$ とのコストの差は

$$\begin{aligned} c(\tilde{t}_{opt},x,y,u) - c(t_{opt},x,y,u) &= c(\tilde{t}_{opt},x,y,u) - \tilde{c}(\tilde{t}_{opt},x,y,u) \\ &+\tilde{c}(\tilde{t}_{opt},x,y,u) - \tilde{c}(t_{opt},x,y,u) + \tilde{c}(t_{opt},x,y,u) - c(t_{opt},x,y,u) \\ &\leq c(\tilde{t}_{opt},x,y,u) - \tilde{c}(\tilde{t}_{opt},x,y,u) + \tilde{c}(t_{opt},x,y,u) - c(t_{opt},x,y,u) \end{aligned}$$

のように表現できる．すなわち，$\tilde{c}$ が一様に $c$ のよい近似であれば，よいチューニングパラメタが選択されることが保証される．

第 3 に，コスト推定関数を参照してチューニングパラメタ制御関数が定義される方式を考える（C2P2）．この場合は実施時のコストの期待値の近似値を

$$\tilde{C}(\tilde{t}) = \iint \tilde{c}(\tilde{t}(x,y),x,y)p_x(x)p_y(y)dxdy$$

と定義し，これを最小にする $\tilde{t} \in T$ を求める．この問題は

(1) コストを推定する $\tilde{c}(t,x,y)$ をどのようにして求めるか

(2) 推定コスト $\tilde{C}_t(\tilde{t})$ を最小にする $\tilde{t} \in T$ をどのようにして決定するか

という 2 つの問題に分解できる．C2P0 の (2) は実施時の問題であったが，C2P2 の (2) は実施前の問題となる．もし C2P0 の最適解 $\tilde{t}_{opt}$ が $T$ に含まれているなら，実施時の平均コストは C2P0 と同じとなる．もし $\tilde{t}_{opt}$ が $T$ に含まれないならば，C2P0 に比べてさらに $C(\tilde{t}) - C(\tilde{t}_{opt})$ だけロスが大きい．なお，$\tilde{c}$ が不適切に計算されるが，$T$ の範囲が適切に制約されているために，C2P0 よりも C2P2 のほうがよい結果を出すということはありうる．

### 4.3 Bayes 統計によるコスト関数推定

次に，C2P0 および C2P2 の部分問題 (1)，すなわち，コストを推定する $\tilde{c}(t,x,y)$ をどのように構成するかにつき，Bayes 統計の枠組みを用いた手法を提案する．

簡単のため，擾乱 $u$ のコストに与える影響は既知で加法的とする．より具体的に，コストの期待値を

$$\bar{c}(t,x,y) = \int c(t,x,y,u)p_u(u)du$$

とおいたとき，コストの分布は

$c(t,x,y,u) = \bar{c}(t,x,y) + u, \qquad u \sim p_u(u)$

であらわされるとする．ただしこの仮定は式を多少具体的にするだけで，以下の議論で本質的ではない．

コストの期待値 $\bar{c}$ の事前情報を Bayes 的な事前分布の形で $p(\bar{c})$ とあらわす．第 $i$ 回目のオフライン試行は，チューニングパラメタ $t_i$，制御される条件 $x_i$, 観測される条件 $y_i$ のもとでおこなわれ，コスト $c_i$ を得たとする．上述の擾乱に対する仮定より，

$p(c_i \mid t_i,x_i,y_i,\bar{c}_i) = p_u(c_i - \bar{c}_i(t_i,x_i,y_i))$

が得られる．これを繰り返しすことで

$$p(\boldsymbol{c} \mid \boldsymbol{t},\boldsymbol{x},\boldsymbol{y},\bar{c}) = \prod p(c_i \mid t_i,x_i,y_i,\bar{c}_i) = \prod p_u(c_i - \bar{c}_i(t_i,x_i,y_i))$$

が得られる．Bayes の公式から

$p(\boldsymbol{c} \mid \boldsymbol{t},\boldsymbol{x},\boldsymbol{y},\bar{c})p(\boldsymbol{t},\boldsymbol{x},\boldsymbol{y} \mid \bar{c})p(\bar{c}) = p(\boldsymbol{c},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y},\bar{c}) = p(\bar{c} \mid \boldsymbol{c},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y})p(\boldsymbol{c},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y})$

となる．ここで $p(\boldsymbol{c},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y})$ は $\bar{c}$ によらない定数であり，また $p(\boldsymbol{t},\boldsymbol{x},\boldsymbol{y} \mid \bar{c})$ も $\bar{c}$ にはよらないと仮定する．このとき，

$p(\bar{c} \mid \boldsymbol{c},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y}) \propto p(\bar{c})p(\boldsymbol{c} \mid \boldsymbol{t},\boldsymbol{x},\boldsymbol{y},\bar{c})$

が得られる（Bayes の定理）．この $p(\bar{c} \mid \boldsymbol{c},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y})$ がコスト推定関数 $\bar{c}$ の事後分布であり，どのような $\bar{c}$ がどの程度もっともらしいかということを具体的かつ定量的に示している．

これより，コストの期待値 $\bar{c}$ の Bayes 的期待値としてコスト推定関数

$$\tilde{c}(t,x,y) = \int \bar{c}(t,x,y)p(\bar{c} \mid \boldsymbol{c},\boldsymbol{t},\boldsymbol{x},\boldsymbol{y})d\bar{c}$$

が得られる．コストの期待値 $\bar{C}$ が

$$\bar{C}(\tilde{t}) = \iiint \bar{c}(\tilde{t}, x, y) p(\bar{c} \mid \boldsymbol{c}, \boldsymbol{t}, \boldsymbol{x}, \boldsymbol{y}) p_x(x) p_y(y) dx dy d\bar{c}$$

で定義されるので，これを最小にするようにチューニングパラメタの選択 $\tilde{t}$ を決定するべきである．

上記の結果は擾乱が既知としている．擾乱の影響が未知の場合には，擾乱の影響をモデル化し，やはり Bayes 統計で表現することが可能である．

### 4.4 最適実験計画

ここは上記の Bayes モデルに基づき，C2P0 を仮定して，実験計画について論ずる．ただし，以下の議論では制御可能パラメタ $x$，観測可能パラメタ $y$ は扱われていない．また，事前分布 $p(\bar{c})$ は「正しい」と仮定する（実験計画の手法をこの事前分布に従う重み付け平均で評価することを意味する）．また，目的関数はコスト総和による定式化とし，具体的には

$$Z = \sum_i c_i + \kappa C(\tilde{t}_{opt})$$

を最小化するものとする．ここで $c_i$ は第 $i$ 回の試行コスト，$\kappa$ は定数， $C(\tilde{t}_{opt})$ は将来の実施コストの期待値である．また，実験計画は**戦略**と呼ばれ，$i$ 回目の試行においてどの選択肢を選ぶかとともに，何回で試行を終了するか（**停止時**，実験結果に依存する）を決定する．

このとき上記のコスト（リスク） $Z$ を最小化する戦略を求める問題は，**逐次統計的最適化**の定式化のひとつである**逐次サンプリング**の問題に帰着される．たとえば文献 2) では第 6 章においてこの問題が取り上げられ，数学的な最適解が示されている．以下ではその概略を示す．

チューニングパラメタに代入できる選択肢は $d$ 個とする．第 $j$ 選択肢のコストの期待値を $\bar{c}_j$ とする．ベクトル $(\bar{c}_1, \ldots, \bar{c}_d) = \bar{c}$ とする．

コスト推定関数には事前分布 $p(\bar{c})$ が与えられている．前小節で論じたように，Bayes 推定を用いることで，事前分布と試行 $\boldsymbol{t}$ と観測値 $\boldsymbol{c}$ から事後分布 $p(\bar{c} \mid \boldsymbol{c}, \boldsymbol{t})$ が得られる．観測値の取りうる範囲を $\Gamma$ とする．すなわち $\boldsymbol{c} \in \Gamma$ であり，任意回の試行で観測される観測値がすべて $\Gamma$ に入っている．このとき，$\bar{c}$ の事後分布の取りうる範囲を $\Lambda$ とする．およそ $\Lambda = \{p(\bar{c} \mid \boldsymbol{c}, \boldsymbol{t}) \mid \boldsymbol{c} \in \Gamma, \boldsymbol{t} \in$ 試行列全体の集合 $\}$ と考えればよい．試行列 $\boldsymbol{t}$ の結果 $\boldsymbol{c}$ により得られた情報は事後分布 $p(\bar{c} \mid \boldsymbol{c}, \boldsymbol{t})$ に凝縮されており，$\Lambda$ のなかのある要素 $\lambda$ はこの事後分布に対応している．すなわち $\lambda$ をひとつ選ぶことは，試行列とその結果の組（同値なものがあればその同値類）をひとつ選ぶことに対応している．なお，文献 2) では $\Lambda$ を有限集合と仮定している．

$\Lambda$ からひとつの要素 $\lambda$ を選んだとき，それが示す $\bar{c}$ の事後分布を $p_\lambda(\bar{c})$ とする．このとき，コスト推定関数 $\tilde{c}_j$ が得られるから，それを最小にする $j$ が（実施コストの意味で最適な）チューニングパラメタの値 $\tilde{t}_{opt}(\lambda)$ となる．事後分布 $\lambda$ に対して，この $\tilde{t}_{opt}(\lambda)$ を選択したときの実施コストの期待値を $g(\lambda)$ とする．すなわち

$$g(\lambda) = \int \bar{C}(\tilde{t}_{opt}(\lambda)) p_\lambda(\bar{c}) d\bar{c}$$

である．また，$\lambda$ に対して，第 $j$ 選択肢のコストの期待値は

$$\bar{c}_j(\lambda) = \int \bar{c}_j p_\lambda(\bar{c}_j) d\bar{c}_j$$

と計算することができる．

事後分布 $\lambda$ と未来の試行列 $\boldsymbol{t}$ に対して，

$$X_{\boldsymbol{t}}^{\lambda} = \sum_{i=1}^{|\boldsymbol{t}|} \bar{c}_{t_i}(\lambda^{\boldsymbol{t}}) + g(\lambda^{\boldsymbol{t}})$$

とおく．ただし $|\boldsymbol{t}|$ は試行列 $\boldsymbol{t}$ の長さ（試行回数），$t_i$ は $\boldsymbol{t}$ における第 $i$ 回試行での選択肢である．また，$\lambda^{\boldsymbol{t}}$ は試行 $\boldsymbol{t}$ の後における事後分布である．$X_{\boldsymbol{t}}^{\lambda}$ は試行列 $\boldsymbol{t}$ に対する目的関数 $Z$ の期待値である．

事後分布 $\lambda$ が与えられているとする．このとき，あらゆる長さのあらゆる試行列 $\boldsymbol{t}$ に対して $X_{\boldsymbol{t}}^{\lambda}$ を計算することができるが，このなかに $\inf_{\boldsymbol{t}} X_{\boldsymbol{t}}^{\lambda}$ を与える $\boldsymbol{t}$ が存在することが証明できる（Snell's envelope）．これを $\boldsymbol{t}_{opt}$ とする．もし $\boldsymbol{t}_{opt}$ が空（試行をしない）であれば，これ以上試行をせず，実施において $\tilde{t}_{opt}(\lambda)$ を選択するのが最適である．もし $\boldsymbol{t}_{opt}$ が空でなければ，試行を継続し，次の試行ではその最初の要素 $t_1$ をチューニングパラメタに選択するのが最適である．

以上が逐次サンプリングの最適解である．しかし，$\boldsymbol{t}$ として無限個の試行列を仮定して比較することが必要であり，特殊な例でないかぎり有限回の計算で求めることはできない．この最適解は動的計画法により構成的に表現することもできる（文献 2) では 6.4 節に説明されている）．これは試行回数が $n$ 以下という制約を課すいわば**試行回数制約による定式化**を行い，試行回数 $n$ が $\infty$ に近づく極限を考えることに相当している．よって，やはり一般的に有限回の計算で求めることができる形にはなっていない．

ちなみに，文献 2) で論じられている他の定式化は次のようなものである．7 章では，コ

ストが過去の選択肢の履歴に依存する場合（**逐次制御**）が論じられている．8 章はオンライン自動チューニングに相当する **multi-armed bandit problem** に充てられている．9 章で論じられている問題は multi-armed bandit problem で，コストがマルコフ連鎖で定まる場合である．試行コスト制約による定式化や，実施コスト制約による定式化に関しては論じられていない．

## 5. ワンステップ近似

### 5.1 提案手法

前節では，C2P0 を仮定し，制御される特徴量 $x$ および観測される特徴量 $y$ がない場合について，コスト総和による定式化における最適実験計画を示した．ただし最適実験計画は有限の計算で求められる形になっていない．そこで，準最適な実験計画を与える手法を提案する．これを**ワンステップ近似**と呼ぶことにする．

ワンステップ近似では，$X^{\lambda}_{\boldsymbol{t}}$ の定義をそのまま使うが，試行列として長さが 0 または 1 のもののみを考える．すなわち，$\boldsymbol{t} = (), (1), (2), \ldots, (d)$ の $d+1$ 通りを考える．これらに対して $X^{\lambda}_{\boldsymbol{t}}$ を計算し，それを最小にする $\boldsymbol{t}$ を選択する．

このとき，$|\boldsymbol{t}| = 1$ に対する $X^{\lambda}_{\boldsymbol{t}}$ の計算式は，オンライン自動チューニング[1] で提案された同様の近似における計算式とほぼ同じである．違いは，オフライン自動チューニングにおける係数 $\kappa$ が，オンライン自動チューニングでは残りの反復回数 $k-1$ に置き換わっている点である．

式がほぼ同じであっても，オフライン自動チューニングとオンライン自動チューニングでは大きな違いがある．それはオフライン自動チューニングでは「試行をやめる」という選択肢が存在することである．これに対して，オンライン自動チューニングでは実施の回数は問題の仮定の一部として与えられており，実験計画により変わることはない．このため，全選択肢のコストに定数を加えてもオンライン自動チューニングでは結果が変わらない．これに対して，オフライン自動チューニングではコストの定数部分が「1 回の試行のコスト」に影響し，いつ試行をやめるかという判断に大きな影響を与える．

### 5.2 予備評価

以下では，提案したワンステップ近似によるオフライン自動チューニングの手法について，シミュレーションによる予備評価を行う．ここでシミュレーションとは乱数を用いた仮想的な自動チューニング問題を提案手法で解くことを示し，現実的な自動チューニング問題を解いたのではないことを示す．

シミュレーションした自動チューニング問題の設定は以下のようである．選択肢は $d = 20$ とし，第 $i$ 選択肢の真のコスト期待値 $c_i = 1 + i/19 + \tau e_i$，その第 $j$ 回目の観測で観測されるコストを $c_i + \sigma e_{ij}$ とした．ここで $\sigma^2 = \tau^2 = 10^{-2}$ とし，$e_i$ および $e_{ij}$ は N(0,1) に従う乱数である．また，オンライン自動チューニングの手法[1] と同様に，$\alpha + \beta i$ という線形モデルを仮定した．最初の 3 回の試行は線形モデルを推定するためにランダムな選択肢を用い，4 回目以降の試行について提案手法で選択肢を決定した．

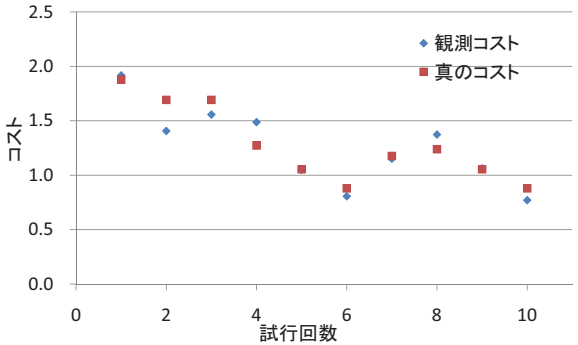


**図 1** オフライン自動チューニングの様子（$d = 20$, $\sigma^2 = \tau^2 = 10^{-2}$, $\kappa = 10^6$）
Fig. 1 A sample of offline automatic tuning

**図 1** に，提案手法によるオフライン自動チューニングの例を示す．最初の 3 回はランダムであるが，残り 7 回でめぼしい選択肢を評価し，わずか 10 回の試行回数で終了した．このとき，コスト最小の選択肢が選ばれた．

**表 2** 実験結果（$d = 20$, $\sigma^2 = \tau^2 = 10^{-2}$，20 回の平均）
Table 2 Experimental results

| $\kappa$ | 平均試行回数 | 平均試行コスト | 平均ロス | 平均ロス $\times\kappa$ |
|---|---|---|---|---|
| $10^3$ | 10.0 | 10.9 | 0.0139 | 13.9 |
| $10^6$ | 19.4 | 19.2 | 0 | 0 |

次に，$\kappa$ を $10^3$ および $10^6$ に設定して，自動チューニングのシミュレーションを 20 回繰り返した結果を**表 2** に示す．ここで**ロス**とは，自動チューニングにより最適と考えられた選択肢 $\tilde{t}_{opt}$ と真のコスト最小の選択肢 $t_{opt}$ の真のコスト差 $c_{\tilde{t}_{opt}} - c_{t_{opt}}$ である．$\kappa = 10^3$

としたときには，平均試行回数が 10 回ちょうどで，平均試行コストが 10.9，平均ロスが 0.0139 であった．目的関数の値は $10.9 + 10^3 \times 0.0139 = 24.8$ であった．試行コストと重みつきロスとがほぼ均衡していることから，ほぼ最適な自動チューニングが達成されたと考えられる．また，$\kappa = 10^6$ としたときには平均試行回数が 19.4 と多くなったが，20 回すべてでコスト最小の選択肢が選ばれた．これも目的関数に従ったほぼ最適な試行が行われたと考えられる．

## 6. おわりに

### 6.1 まとめ

本稿では，オフライン自動チューニングの数理手法について考察した．

著者らはこれまでオンライン自動チューニングについて研究してきたが，今回はオフライン自動チューニングを視野に入れて，自動チューニングの基礎概念について再考した．オフライン自動チューニングにおいては実施における実行条件を予想して試行時の実行条件を与えることが重要であるが，静的な条件と動的な条件，観測できる条件と観測できない条件，制御できる条件と制御できない条件，特徴量と非特徴量という視点で実行条件を分類・整理した．これらに対応して，定数的適応と関数的適応が考えられる．

次に，自動チューニングの関数的適応について，チューニングパラメタ制御関数とコスト推定関数の 2 つの方法を示し，それぞれについて開発者により組み込まれる場合と試行実験により推定される場合を考えて，6 通りの関数的適応のあり方を示した．すなわち，開発者により関数的適応が組み込まれる C0P1, C1P0, C1P2 と，試行実験により推定される C0P2, C2P0, C2P2 である．

また，オフライン自動チューニングにおける試行コストと実施コストの 2 つの目的関数の組み合わせ方について考察した．これらの基礎的な考察に基づき，（オンライン自動チューニングではなく）オフライン自動チューニングが必要となる状況，オフライン・オンライン混合自動チューニングが必要となる状況について考察した．

さらに，非特徴量について仮定を加え，C0P2, C2P0, C2P2 における実施コストの推定方法について論じた．ここでは，C0P2 に課題が見つかった．次に，C2P0 を主に想定して Bayes 統計によるコスト推定関数の構築方法を論じた．続いて，コスト総和による定式化において Bayes 統計による最適実験計画（既知）を紹介した．ただしこれは計算量的には現実的ではない．

そこで，準最適な実験計画手法として，ワンステップ近似を提案した．提案手法について，シミュレーションによる予備評価を行い，試行コストと実施コストの重みつき和が近似的に最小化されていることを確認した．

### 6.2 今後の課題

本稿ではオフライン自動チューニングのモデルと数理について検討した．既存の自動チューニング技術の実装の多くがオフライン自動チューニングであるが，その数理的な定式化と扱いについては曖昧なままであることが多く，今後の研究課題は多い．以下，本稿と直接関係のある課題を列挙する．

- ワンステップ近似の詳細なシミュレーションによる評価
- C2P0 と C2P2 の差の解明
- 制御される特徴量 $x$, 観測される特徴量 $y$ の数理的扱い
- 中断のための数理モデルと最適な中断手法の開発
- 問題サイズの縮小のための数理モデルと最適な問題サイズ決定手法の開発
- オフライン・オンライン混合自動チューニングの定式化と数理手法の開発
- 実際の自動チューニング問題への適用

なかでも，制御される特徴量と観測される特徴量を扱う数理モデルと数理手法を検討することが根本的に重要である．

これらに加えて，既存のオフライン自動チューニング技術を数理的な視点でとらえなおして知見を整理することが求められる．また，数理的な定式化をシステムとして実体化し，自動チューニングの効率化に役立てることが必要である．

**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「マルチコア複合環境を指向した適応型自動チューニング技術」の支援を受けています．

## 参考文献

1) Suda, R.: A Bayesian Method for Online Code Selection: Toward Efficient and Robust Methods of Automatic Tuning, *Proc. iWAPT 2007*, pp.23–31 (2007).
2) Cailori, R. and Dalang, R.C.: *Sequential Stochastic Optimization*, John Wiley and Sons, Inc. (1996).