Vol.2021-HPC-178 No.16
2021/3/16

GPU クラスタを用いて並列化した自動チューニングの機械学習プログラムへの適用と安定性の検証

藤家空太郎[†1]　多部田敏樹[†1]　藤井昭宏[†1]　田中輝雄[†1]
加藤由花[†2]　大島聡史[†3]　片桐孝洋[†3]

概要：我々は反復一次元探索を用いた自動チューニングによる複数パラメータ同時推定の研究に取り組んでおり，今回それを機械学習プログラムに適用した．機械学習プログラムは測定ごとに訓練データが変わり，同じハイパーパラメータで複数回実行しても結果にばらつきがある．本報告は歩行者経路予測の機械学習プログラムに適用した自動チューニングを GPU クラスタで並列化することで，逐次実行で約 11 日かかる推定が約 12 時間で完了することを示す．また，推定したハイパーパラメータを同じ GPU クラスタで複数回同時に測定した結果の平均と分布から，推定したハイパーパラメータが現在経験則で良いとされ使われているパラメータと比較して優位性があり，反復一次元探索を用いた自動チューニングが結果にばらつきのあるプログラムでも安定して推定が行えていることを示す．

Application and Stability Verification of Parallelized Automatic Tuning to Machine Learning Programs on GPU Cluster

SORATARO FUJIKA[†1]　TOSHIKI TABETA[†1]　AKIHIRO FUJII[†1]
TERUO TANAKA[†1]　YUKA KATO[†2]　SATOSHI OHSHIMA[†3]
TAKAHIRO KATAGIRI[†3]

Abstract: We are working on the study of simultaneous estimation of multiple parameters by automatic tuning using iterative one-dimensional search, and this time we applied it to a machine learning program. In the machine learning program, the training data changes for each measurement, and the results vary even if the same hyperparameters are measured multiple times. This report shows that by parallelizing the automatic tuning applied to the machine learning program for pedestrian route prediction in a GPU cluster, the estimation that takes about 11 days for sequential execution can be completed in about 12 hours. In addition, from the average and distribution of the estimated hyperparameters measured multiple times in the same GPU cluster at the same time, it can be seen that the estimated hyperparameters are superior to the hyperparameters currently used according to the rule of thumb. From the above, it is shown that automatic tuning using iterative one-dimensional search enables stable estimation even in programs with varying results.

1. はじめに

コンピュータの性能向上に伴い，コンピュータアーキテクチャは複雑化してきている．そのため，ユーザが実行環境のコンピュータアーキテクチャや性能パラメータの特性を理解し，手動で適切な性能パラメータの設定をすることは困難である．そこで，自動で性能パラメータの最適化を行う技術として自動チューニングの研究が進められている[1][2][3]．性能パラメータとはチューニングで最適化する値（消費電力，実行時間等）に影響を及ぼすパラメータであり，ユーザが実行するプログラム（ユーザプログラム）に組み込み，調整することで，性能向上を図ることができる．最適な性能パラメータの値は実行環境によって異なるため，実行環境に合わせた性能パラメータの設定は重要である．

複数の性能パラメータからなる空間をパラメータ空間とする．我々は，複数の性能パラメータを同時に推定する手法として，多次元のパラメータ空間における線形探索を繰り返し行う自動チューニング機構を提案している[4][5]．この手法は，性能パラメータの取りうる値の組み合わせの設定とユーザプログラムの実行による実行性能の取得を繰り返して，性能パラメータの最適な組合せを自動的に推定する．本研究ではこの自動チューニング機構を，機械学習プログラムに適用し，実行性能に影響を及ぼすハイパーパラメータの最適化を行う．本研究で用いる機械学習プログラムは実測 1 回に 30 分を要するため，我々の自動チューニング機構でも推定完了に数日かかる．また，この機械学習プログラムは実行ごとに訓練データが変わるため，同じハイパーパラメータを用いて複数回実測すると，結果にばらつきが生じる．自動チューニングではこのばらつきの影響を受ける可能性がある．

本研究の目的は以下の 2 つである．第一の目的は自動チ

†1 工学院大学
Kogakuin University
†2 東京女子大学
Tokyo Woman’s Christian University
†3 名古屋大学
Nagoya University

ューニング機構が推定する際に必要な実測を，並列化して推定時間を削減する. 並列化には機械学習に有効な GPU を複数利用できる名古屋大学のスーパーコンピュータ「不老」Type II サブシステム[6]を用いる．第 2 の目的は，ばらつきが存在するプログラムに対しても，自動チューニング機構で求めた結果が有効であるか確かめることである．先行研究として，追加測定を用いてばらつきに考慮した自動チューニングの研究がある[7]．そこで推定結果から性能の良かったハイパーパラメータの組み合わせを複数回追加測定し，その平均と分布をもとに確認する．本報告では自動チューニング機構を用いた機械学習プログラムのハイパーパラメータ推定を推定フェーズ，自動チューニング機構に対するばらつきの影響の検証を検証フェーズとする．

2. 自動チューニング機構の推定手法

2.1 推定手法の概要

我々の自動チューニング機構では，パラメータ推定に反復一次元探索を用いる．反復一次元探索は標本逐次追加型性能パラメータ推定法（IPPE）による一次元探索を，探索方向を増やしながら繰り返し行う手法である．

IPPE はスプライン型の近似関数 d-Spline を利用したパラメータの最適値推定法である．d-Spline は微分連続性を用いない離散型の近似関数で，実測データの動きに柔軟に追随し，少ない標本点でも安定して解が得られ，計算量が少ないという特徴がある[8][9]．IPPE の最適値推定の手順を図 1 に示す．初期パラメータは探索を行う領域を均等に分ける点をいくつか選択する．今回の場合は，全パターンの両端を含む等間隔の 4 点である．次に選択された標本点を実測する．実測データから d-Spline を計算し最適値を評価する． 終了条件を満たしていない場合，次の標本点の選択は d-Spline が最適値になる点を選択する．ただし，選択される点が実測済みの場合は，未実測の点のうち二階差分が最大の点を選択する．

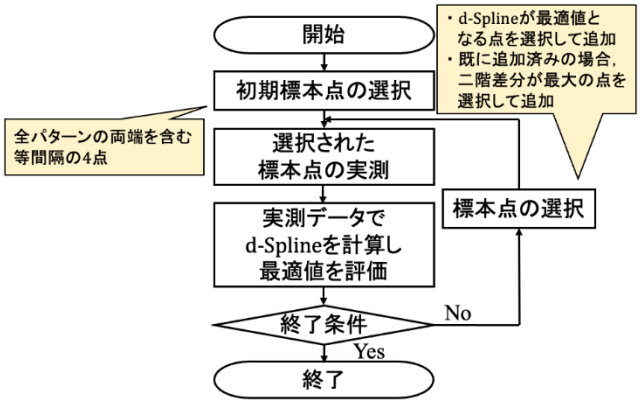

図 1　IPPE の最適値推定の手順．標本点の選択と実測，d-Spline の計算を繰り返し推定する．

2.2 反復一次元探索

反復一次元探索は一次元の d-Spline を用いて複数の性能パラメータを同時に推定する手法である．

我々の自動チューニング機構において，反復一次元探索を用いて n 個の性能パラメータを推定する手順を，図 2 に示す．初めに初期パラメータを基準点とする．以降，探索によって最適値に選ばれた点が新たな基準点となる．探索方向の選択では，基準点に隣接する点を比較して探索方向を決定する．これを方向探索とする．次に一次元 d-Spline を計算し，最適値となる点を推定する．これを一次元探索とする．各探索において，一度探索した点は再度実測を行わない．方向探索と一次元探索を基準点が同点になる（連続して同じ点が選ばれる）まで繰り返す．基準点が同点となった場合，探索する軸を増やし，再び方向探索と一次元探索を繰り返す．探索方向は 1st step から始まり，探索する軸を増やすごとに 2nd step, 3rd step, …, nth step と変化する．1st step は性能パラメータの各軸方向であり，2nd step は 2 軸を含む斜め方向，3rd step は 3 軸を含む斜め方向と軸が増えていく．nth step で n 軸を含む斜め方向（全方向）の探索を行い, 基準点が同点となれば, 反復一次元探索を終了する．

n 個の性能パラメータを探索する場合，次元数は n となり, 探索方向のパターンは式 1 で表せる．ここでkは step 数であり$1 \le k \le n$である．

$$\sum_{d=0}^{n-1}\sum_{s=0}^{k-1}\binom{d}{s}2^s \qquad (1)$$

6 次元の場合，性能パラメータの各軸方向の探索は 6 パターンだが，全方向の探索は 364 パターンに及ぶ．探索方向を順に増やしていくことにより推定の精度をほぼ変えずに実測回数を減らすことができる．

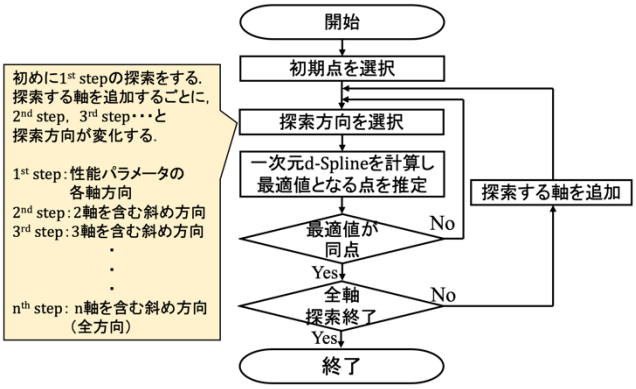

図 2　反復一次元探索を用いて n 個の性能パラメータを推定する手順．初めから全方向を調べずに，徐々に探索範囲を広げていく．

3. 並列化と自動チューニング機構の実装

3.1 並列化の概要

自動チューニング機構を機械学習プログラムに組み込み，ハイパーパラメータの最適化を行う．本報告ではこのハイパーパラメータの推定を推定フェーズと呼ぶ．ハイパーパラメータは学習率やドロップアウト率のような機械学習の挙動を制御するパラメータである．推論や予測などにおいて自動で設定されず，手動で設定する必要があるが，経験則で調整するのは難しく，最適値を求めるには膨大な時間がかかる．

従来の自動チューニング機構（逐次実行）をそのまま組み込んだ場合，学習に時間のかかる機械学習プログラムのハイパーパラメータ推定は数日かかる．今回の場合 1 回の学習に 30 分かかる機械学習プログラムに自動チューニングを適用した．そこで方向探索と一次元探索の実測を並列化することで実行時間の削減を行う．

3.2 自動チューニング機構の並列化

複数のジョブを並列に実行できる環境に向けて，自動チューニング機構を改良する．図 3 に示すように，並列化により反復一次元探索における方向探索と一次元探索の実測をまとめて行う．

実行環境には名古屋大学のスーパーコンピュータである「不老」Type II サブシステムを用いる．リソースグループは cx-share とする．cx-share は 1 ユーザあたり最大 50 のジョブを同時に実行できる．1 ジョブにつき 1GPU を使うことができ，使用するノードはユーザ間で共有する．

並列化は複数のジョブで同時に実測することで行う．図 4 に探索する点の決定後の自動チューニング機構の状態の遷移を示す．自動チューニング機構は状態「ジョブ投入」「ジョブ完了確認」「データ整理」を持ち，それぞれの完了を確認してから次の状態に移る．

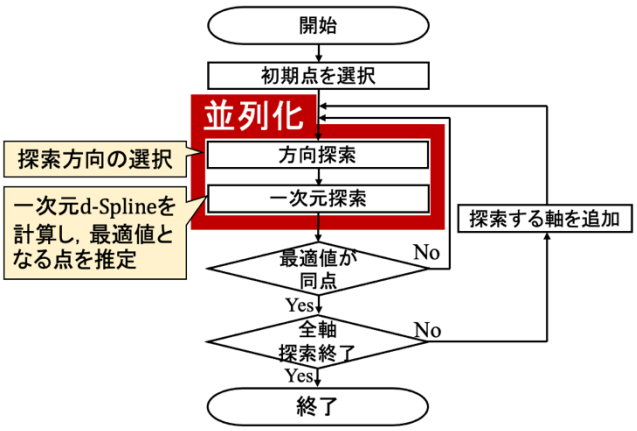

図 3 反復一次元探索の並列化のイメージ．方向探索と一次元探索の実測をまとめて行うことで，実行時間を大きく削減する．

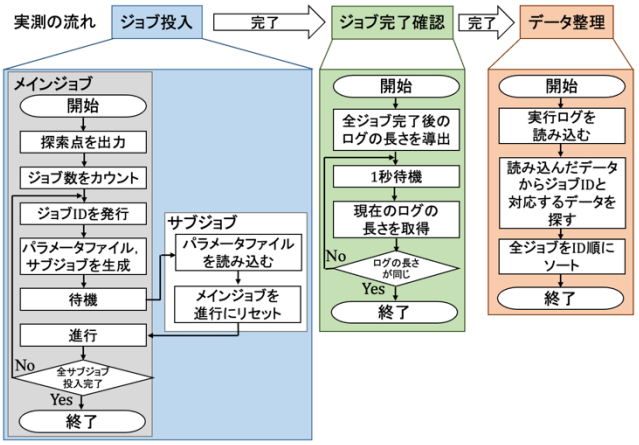

図 4 探索する点の決定後の自動チューニング機構の状態の遷移．メインジョブはサブジョブがすべて完了したか同期を取って確認する必要がある．データ整理完了後は，従来手法と同様に基準点が移動したかを確認する．

ジョブ投入では，メインジョブが並列に実測するハイパーパラメータの組み合わせの数だけサブジョブを生成する．各サブジョブは実測するハイパーパラメータが記述された csv のファイルを受け取り，そのハイパーパラメータを設定して実測する．メインジョブは一つひとつのサブジョブが実行されるのを確認しながら次のサブジョブを生成する．

ジョブ完了確認で全てのサブジョブが実測を終えたかを確認する．各サブジョブは実測終了後，実測結果（ログ）を共通の csv ファイルに記録する．メインジョブは 1 秒ごとに csv ファイルに記録されたログの長さを確認する．式 2 で求められる全ジョブ完了後のログの長さと比較して，現在のログの長さが同じになっている場合，ジョブ完了確認を終了する．

$$（投入前のログの長さ）\times（投入したサブジョブ数）\tag{2}$$

データ整理では，ジョブが投入された順に完了するとは限らないため，ジョブ，パラメータ，実行性能データの対応付けを行う．

データ整理終了後は，従来手法と同様に基準点が移動するかを確認する．移動しなかった場合は探索する軸を追加し，再び方向探索と一次元探索を同時に行う．

方向探索と一次元探索を同時に実測するため，探索する方向が増加するほど探索する点が増え，並列度が上がることとなる．しかし，今回のリソースグループではジョブの同時実行可能数が 50 のため，最大で 50 の点を同時に実測することになる．

4. 評価対象プログラムへの適用

4.1 評価対象プログラム

提案手法を，機械学習を用いた歩行者経路予測アプリケーション[10]に適用する．このアプリケーションは学習フェーズと予測フェーズによって，ターゲットとなる人の過去の移動軌跡データから，歩行者の未来の経路と到着地点を予測する．

学習フェーズでは，データセットよりランダムに選んだ画像から歩行者の位置情報を抽出し，構成した訓練データを用いて予測器を生成する．予測器の生成には，再帰型ニューラルネットワーク（RNN）ベースの手法であるSocial LSTMを用いる[11]．Social LSTMは、同一の時間と軌道上で発生する人と人の相互作用を自動的に学習する．

予測フェーズはLiDARで歩行者の移動の軌跡を計測し，計測データから予測器への入力データを生成して，予測器で経路予測を行う．

評価指標値には実際の歩行者の到達地点と予測器が予測した到達地点の誤差であるFDE（Final Displacement Error）を用いる．FDEが小さいほど，精度がよいことになる．本研究ではFDEが小さくなるハイパーパラメータの組み合わせを推定する．

4.2 自動チューニング機構の適用

図5に自動チューニング機構と歩行者経路予測アプリケーションのデータの位置付けを示す．自動チューニング機構は予測器の生成に用いる機械学習のハイパーパラメータを最適化する．自動チューニング機構は，設定されたハイパーパラメータを用いて予測器で実測を行い，性能評価値（今回の場合はFDE）取得する．この時最初に用いたハイパーパラメータを初期ハイパーパラメータと呼ぶ．自動チューニング機構は取得した性能評価値から反復一次元探索によって，探索するハイパーパラメータの組み合わせを設定する．予測器は設定されたハイパーパラメータを用いて実測する．この手順を反復一次元探索が終了するまで繰り返す．ここで提案手法では複数のGPUを使うことのできるスーパーコンピュータ上において，複数のハイパーパラメータの組み合わせを並列に実測する．

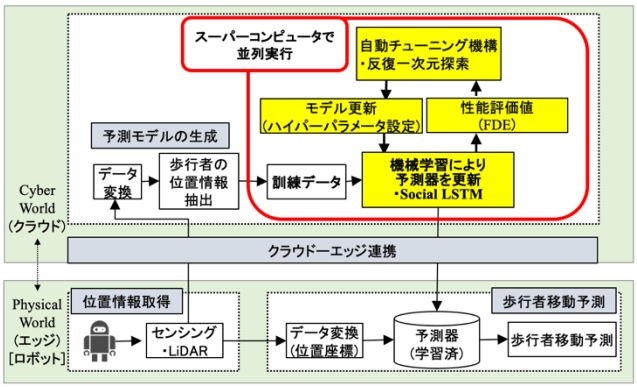

図5　自動チューニング機構と歩行者経路予測アプリケーションの位置付け．自動チューニング機構は予測器の生成に用いる機械学習のハイパーパラメータを最適化する．性能評価値（FDE）の実測は，複数のGPUを使うことで，並列に行う．

4.3 ハイパーパラメータの推定結果

今回推定するハイパーパラメータは 6 種類であり，各ハイパーパラメータは 5 つの値を取り得る．よってハイパーパラメータの組み合わせは全部で 15625 通りとなる．従来手法と提案手法の推定結果を表 1 に示す．初期パラメータは，現在経験則で最良として使われているハイパーパラメータの組み合わせである．推定完了までの時間は，従来手法が約 279 時間（11.6 日）なのに対し，提案手法は約 12.3 時間となり，約 23 倍高速化した．一方で探索回数は従来手法の 553 回に対し，提案手法では 761 回となった．これは提案手法が探索方向すべてを一度に探索するため，探索回数が増加したからと考えられる．評価指標値（FDE）につ

表1　推定フェーズの逐次実行と並列実行の推定結果. 初期パラメータは, 現在経験則で最良として使われているハイパーパラメータの組み合わせ.

		逐次実行	並列実行	初期パラメータ
所要時間		279 時間	12.3 時間	
探索回数		553 回	761 回	
評価指標値（FDE）		0.89m	0.81m	1.21m
ハイパーパラメータ	Rnn size	512	512	128
	Grad clip	9	10	10
	Learning rate	0.003	0.01	0.003
	Decay rate	0.9	0.95	0.95
	Dropout	0.3	0.3	0.5
	Lambda param	0.0002	0.0001	0.0005

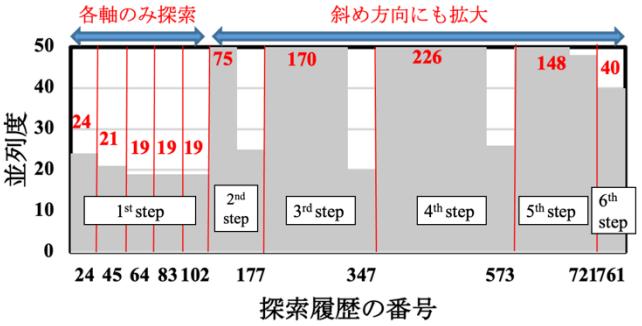

図 6　提案手法の並列度．探索履歴の番号とは何番目に探索した点かを示す．探索方向は初めから全方向を調べずに，段階的に増やしていくことで探索に必要な実測を削減する．

いて推定前は 12.1m となっていたが，今回の評価では従来手法が 0.89m，提案手法が 0.81m と削減できた．

図 6 に提案手法の並列度を示す．縦軸は並列度，横軸はパラメータ推定において何番目に探索した点かを表す探索履歴の番号である．提案手法では，赤線の区切りでそれぞれ並列化されている探索 102 回目までが，1st step であるハイパーパラメータの各軸方向のみで探索しており，探索点が 4 回移動した．探索範囲が小さいため並列度は低く，最大でも 34 である．探索 102 回目以降は，複数のパラメータを組み合わせた斜め方向にも拡大し，探索点が移動しなかった．探索範囲が大きくなったため 50 以上の並列度を抽出した．そのため 4th step の場合，226 並列分全て実行する場合には並列単位で 5 回の実行が必要となる．

5. ばらつきの影響の検証

5.1 機械学習プログラムの結果のばらつき

機械学習プログラムは，訓練データを使って学習を行い学習結果から目的となるタスクをこなす．今回用いた歩行者経路予測プログラムの場合，データセットよりランダムに選んだ画像から歩行者の位置情報を抽出し，訓練データを構成している．ここで訓練データを構成する画像がランダムに選ばれることから，実行ごとに訓練データが変化する．そのため，同じハイパーパラメータの組み合わせを用いて複数回実行すると，FDE にばらつきが生じる．これを結果のばらつきと呼ぶことにする．

提案手法を歩行者経路予測アプリケーションに組み込んでハイパーパラメータを推定した結果を FDE が昇順になるよう並び替えて図 7 に示す．縦軸は FDE，横軸はパラメータ推定において何番目に探索した点かを表す探索履歴の番号である（値を省略）．初期パラメータは， 現在経験則で最良として使われているハイパーパラメータの組み合わせである．初期パラメータの FDE は 1.21 と，探索したハイパーパラメータの組み合わせにおいて高くなっている．しかし，今回探索したハイパーパラメータの組み合わせは 15625 通り中 761 通りであり，反復一次元探索によってより最適と考えられるハイパーパラメータを探索している．そのため，初期パラメータが全ハイパーパラメータの組み合わせの中でも特に悪いハイパーパラメータの組み合わせであるとは限らない．

図 7 で示した結果のうち，FDE の小さい上位 30 通りのハイパーパラメータの組み合わせを図 8 に示す．縦軸は FDE，横軸はパラメータ推定において何番目に探索した点かを表す探索履歴の番号である．この結果では，141 回目に探索したハイパーパラメータの組み合わせが，他のハイパーパラメータの組み合わせよりも FDE が小さい．しかし提案手法では各ハイパーパラメータの組み合わせを 1 回しか実測しない．そのためこの結果は機械学習プログラムの結果のばらつきの影響を受けており，再度自動チューニング

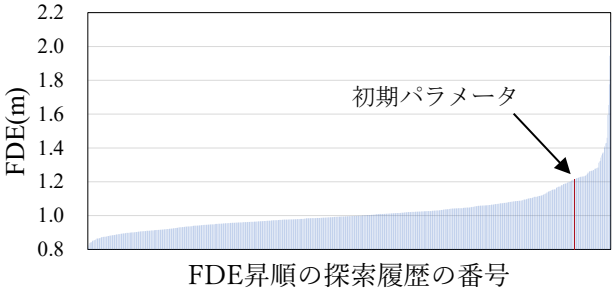

図 7　ハイパーパラメータの推定結果（FDE 昇順）．横軸の探索履歴の番号は，推定フェーズでのパラメータ推定において何番目に探索した点かを示す（値は省略）．初期パラメータは，現在経験則で最良としてわれているハイパーパラメータの組み合わせである．

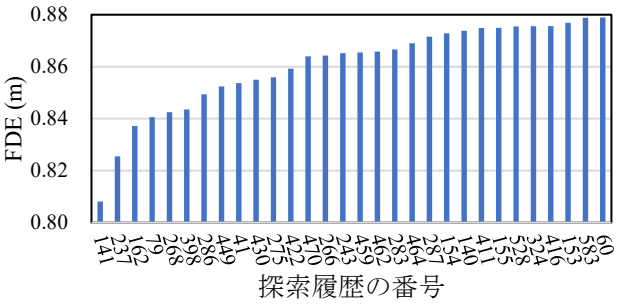

図 8　推定結果のうち，FDE の小さい上位 30 通りのハイパーパラメータの組み合わせ．横軸の探索履歴の番号は，推定フェーズでのパラメータ推定において何番目に探索した点かを示す．

グを行った場合にチューニング結果が大きく変わる可能性が考えられる．

ばらつきを考慮して最適なハイパーパラメータを設定するために，全てのハイパーパラメータの組み合わせを実測する手法が考えられる．しかし，その手法は実測回数が増え，実行時間が膨大になるため現実的ではない．我々の自動チューニング機構では探索する回数を反復一次元探索によって絞ることで，より少ない探索回数での推定を行なっている．そこで推定フェーズで FDE が小さかったハイパーパラメータの組み合わせを追加で測定することで，ばらつきが存在するプログラムに対しても，自動チューニング機構が安定した推定を行えているか調べる．本報告ではこのばらつきの影響の検証を検証フェーズと呼ぶ．

5.2 ばらつきの影響の検証手法

図 9 に検証フェーズの手法を示す．縦軸は性能評価値（今回の場合は FDE），横軸は追加測定の回数である．P1～P5 は提案手法によって推定した，性能評価値の良かった上位 5 つのハイパーパラメータの組み合わせである．まずステップ 1 で最も性能の良かった P1 を並列に複数回同時に実測する（今回は 10 回 ）．実測結果から最大，最小の

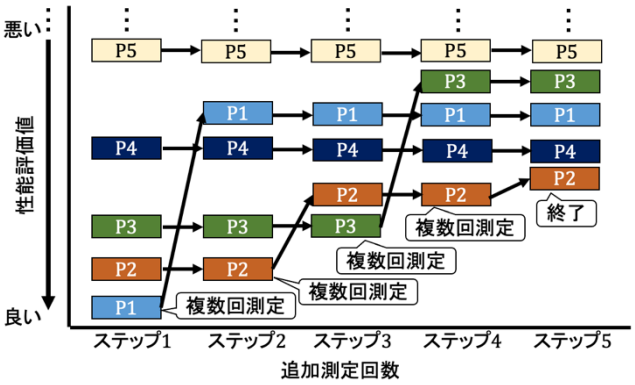

図 9　自動チューニング機構に対するばらつきの影響の検証手法．P1～P5 は提案手法によって推定した，性能評価値の良かった上位 5 つのハイパーパラメータの組み合わせである．

値を除外して平均を求めることでばらつきによって発生する極端なデータを削除して安定した値を取得する．P1 の値を求めた平均値に更新する．P1 の値が大きくなり P2 が最も性能評価値の良い点となる．次にステップ 2 で最も性能の良くなった P2 を測定して値を更新する．以上を繰り返し，最も性能評価値の良い点が変わらなくなるまで値を更新する．ステップ 4 では再び P2 を測定している．ステップ 5 で値の更新後も P2 が変わらず最も性能評価値の良い点であるため，検証フェーズを終了する．

5.3　並列化

検証フェーズも，推定フェーズと同様に並列化し，複数のハイパーパラメータの組み合わせを同時に実測する．1 つのハイパーパラメータの組み合わせにつき 10 回の追加測定する．使用したリソースグループでの同時ジョブ実行回数は 50 回であることから，今回は 5 つのハイパーパラメータの組み合わせを．それぞれ 10 回ずつ同時に追加測定する．

5.4　検証結果

検証フェーズでは推定フェーズで推定したハイパーパラメータの組み合わせを性能評価値の良い（FDE が小さい）順に調べる．今回は 30 通り調べた．図 10 は検証フェーズの結果，FDE の小さかった上位 10 通りと初期パラメータの分布である．縦軸が FDE，凡例にある各箱ひげの値は探索履歴の番号で，推定フェーズでのパラメータ推定において何番目に探索した点かを示す．初期パラメータとは歩行者経路予測アプリケーションで，現在経験則で最良として使われているハイパーパラメータの組み合わせである．各箱ひげの点の分布は同じハイパーパラメータの組み合わせで複数回の実行した結果の変化であり，結果のばらつきのことである．初期パラメータに対して推定した点は FDE が優位な位置で安定していることから，推定フェーズで初期パラメータよりさらに最適なハイパーパラメータの組み合わせを推定できていることが分かる．図 8 と比較した場合，

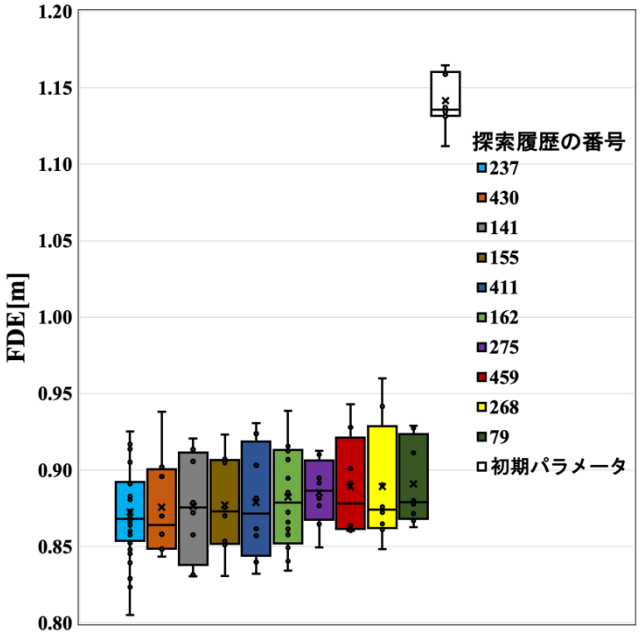

図 10　検証フェーズの結果，FDE の小さかった上位 10 通りと初期パラメータの分布．初期パラメータとは歩行者経路予測アプリケーションで，現在経験則で最良として使われているハイパーパラメータの組み合わせである．

表2　推定フェーズと検証フェーズの推定結果．初期パラメータのFDEは，検証フェーズの実測と同様に，10回の実測結果から最大，最小を除いた平均値．

		推定フェーズ	検証フェーズ	初期パラメータ
所要時間		12. 3 時間	3. 6 時間	
実測回数		761 回	30 回	
評価指標値（FDE）		0.81m	0.87m	1.14m
ハイパーパラメータ	Rnn size	512	512	128
	Grad clip	10	7	10
	Learning rate	0.01	0.01	0.003
	Decay rate	0.975	0.975	0.95
	Dropout	0.3	0.3	0.5
	Lambda param	0.0002	0.0002	0.0005

表 3　推定フェーズと検証フェーズの実行時間．括弧内の数字は，検証フェーズを行うことによる自動チューニング全体の実行時間の増分である．

	推定フェーズ	検証フェーズ（自動チューニング全体の実行時間の増分）
逐次	279 時間	81 時間（29%）
並列	12.3 時間	3.6 時間（29%）

検証フェーズの結果 FDE の小さかった上位 10 通りのハイパーパラメータの組み合わせは，推定フェーズで推定した組み合わせのうち上位 24 までに含まれており，大きく変動していない．また，検証フェーズで最も FDE の小さかったハイパーパラメータの組み合わせは，推定フェーズ 2 番目に FDE の小さかった組み合わせである．これらのことから，提案手法が結果にばらつきのある機械学習プログラムのハイパーパラメータ推定を安定して行えていることが分かる．

表 2 に推定フェーズと検証フェーズの推定結果を示す．初期パラメータの FDE は，検証フェーズの実測と同様に，10 回の実測結果から最大，最小を除いた平均値である．推定フェーズと検証フェーズで最良だったハイパーパラメータの組み合わせのうち，異なるのは Grad clip のみである．このことから 2 つのフェーズの推定結果が非常に近いことが分かる．推定フェーズに比べ検証フェーズでは FDE が大きくなった．これは，検証フェーズが 10 回の実測結果から最大，最小を取り除いた平均を求めたことで，FDE がばらつきによって発生した極端な値の影響を受けていないためである．

表 3 に推定フェーズと検証フェーズの実行時間を示す．並列化した検証フェーズの実行時間は 3.6 時間となり，逐次実行よりも実行時間を削減している．検証フェーズは 50 並列で実行したが，逐次実行から実行時間が 50 倍に減少はしなかった．これは並列化によって複数のハイパーパラメータの組み合わせを 5 個同時に実測したために，逐次実行よりも検証フェーズの終了条件（最も性能評価値が良いハイパーパラメータの組み合わせが変わらない）を満たしにくくなり，実測したハイパーパラメータの総数が増加したことが考えられる．ここで推定フェーズと検証フェーズを合わせた自動チューニング全体の実行時間を考える．表の括弧内の数字は，検証フェーズを行うことによる自動チューニングの実行時間の増分を示している．逐次実行の場合，推定フェーズに加えて検証フェーズを行うことで自動チューニング全体の実行時間は 29%増加した．また，並列実行の場合も 29%の増加となった．

6. おわりに

本研究では，従来の反復一次元探索を用いた自動チューニング機構を，複数の GPU が使えるスーパーコンピュータの環境を利用して，並列に実測を行えるよう改良した．改良した自動チューニング機構を機械学習プログラムのハイパーパラメータ推定に適用した．また，改良した自動チューニング機構がばらつきの存在するプログラムに対しても有効であるか，推定したハイパーパラメータを追加で測定することで検証した

本報告の結論は以下の 2 点である．

- 並列化によって，適用した機械学習プログラムのハイパーパラメータ推定にかかる時間を 279 時間(11.6 日)から 12.3 時間まで削減した．
- 各ハイパーパラメータの組み合わせの実測を 1 回しか行わない提案手法でも，安定したハイパーパラメータ推定が行えていることを確認した．

今回は，歩行者経路予測アプリケーションに対して並列化した自動チューニング機構を適用し，ハイパーパラメータ推定を行なった．この自動チューニング機構を用いたハイパーパラメータ推定は今回で 2 件目である．今後の課題として，対象プログラムを変えても，同様に安定した推定が行えるか確認することがあげられる．

謝辞

本研究の一部は JSPS 科研費 JP18K19782, JP18K11340 の助成および名古屋大学 HPC 計算科学連携研究プロジェクトの支援によるものである．

参考文献

[1] 片桐孝洋，ソフトウエア自動チューニング-数値計算ソフトウエアへの適用とその可能性-，慧文社 ISBN4-905849-18-7（2004）．
[2] 黒田久泰，直野健，岩下武史，特集：科学技術計算におけるソフトウェア自動チューニング，学会誌「情報処理」，第 50 巻，情報処理学会（2009）．
[3] J. Bilmes, K. Asanovic, C.W. Chin, J. Demmel, Optimizing Matrix Multiply using PHiPAC: a Portable, High-Performance, ANSI C Coding Methodology, in Proc. the 11 th international conference in Supercomputing, Vol.97, pp.340-347 (1997).
[4] 望月大義，藤井昭宏，田中輝雄，ソフトウェア自動チューニングにおける複数同時性能パラメタ探索手法の提案と評価，情報処理学会論文誌，vol.11，No2，pp.1-16（2018）．
[5] Toshiki Tabeta, Naoto Seki, Akihiro Fuji, Teruo Tanaka, Hiroyuki Takizawa, An Optimization Technology of Software Auto-Tuning Applied to Machine Learning Software, Poster Session on HPCAsia2020 (2020).
[6] 名古屋大学情報連携推進本部，スーパーコンピュータ「不老」紹介， http://www.icts.nagoya-u.ac.jp/ja/sc/overview.html, (最終アクセス 2021.年 1 月 4 日).
[7] 関直人，范谷瑛，多部田敏樹，藤井昭宏，田中輝雄，性能パラメータ推定における評価対象プログラムの実行時間の揺らぎに対応した自動チューニング手法の提案，研究報告ハイパフォーマンスコンピューティング（HPC）,vol.2019-HPC-169，No9，pp.1-8（2019）．
[8] Teruo Tanaka, Ryo Otsuka, Akihiro Fujii Takahiro Katagiri, Toshiyuki Imamura, Implementation of d-Spline-based incremental performance parameter estimation method with ppOpen-AT, Scientific Programming 2014, vol. 22, no. 4, pp. 299-307, (2014).
[9] Teruo Tanaka, Takahiro Katagiri, Toshitsugu Yuda, d-Spline Based Incremental Parameter Estimation in Automatic Performance Tuning, In Proceedings of the 8th International Conference on Applied Parallel Computing: State of the Art in Scientific Computing, LNCS, Springer, Vol. 4699, pp. 986-995, (2007).
[10] Rina Akabane, Yuka Kato, Pedestrian Trajectory Prediction Using Pre-trained Machine Learning Model for Human-Following Mobile Robot, IEEE International Conference on Big Data Workshop (IoTDA 2020), pp.3453-3458, (2020).

[11] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, Silvio Savarese, Social LSTM: Human Trajectory Prediction in Crowded Spaces, Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), pp.961-971 (2016) .