情報処理学会論文誌　コンピューティングシステム　Vol.14　No.2　1–12 (Aug. 2021)

ベイズ最適化による洪水シミュレーションコードの負荷分散自動調整

石塚 歩[1,a)]　山下 毅[2,3,b)]　江川 隆輔[3,4,c)]　滝沢 寛之[3,d)]　山本 道[5,e)]　風間 聡[5,f)]

受付日 2020年10月20日, 採録日 2021年3月19日

概要：MPI を用いた並列計算の実行時間を短縮するためには，各プロセスの実行時間が均等になるよう負荷分散を調整する必要がある．並列計算の一例として全国の洪水シミュレーションを考えると，浸水深の計算が行われる陸地領域の分布によって計算機のベクトル演算性能は変化するため，各プロセスの負荷が均等となるような分割方法を求めることは困難である．また，計算要素ごとに土地利用データに基づく個別の計算が行われていることから，これを考慮した負荷分散調整はさらに困難なものとなる．従来は，事前に静的な調整を行うため，熟練のプログラマが手作業で負荷の均等化を行ってきた．しかし，これには経験と試行錯誤に基づく多大な労力が必要である．シミュレーションには 1 回試行するごとに無視できない実行時間がかかるため，グリッドサーチやランダムサーチといった単純な探索手法による負荷分散の自動化は非現実的である．したがって，本研究では前述の洪水シミュレーションを対象に，ベイズ最適化と探索範囲の調整による静的負荷分散の自動化を提案する．ベイズ最適化とは，ベイズ推定に基づいて未知の関数の最大値を効率的に探索する方法であり，これを用いることでグリッドサーチやランダムサーチに比べて少ない試行回数で負荷分散調整を行うことが可能となる．さらに，洪水シミュレーションの場合，各プロセスが担当する陸地領域の面積から負荷を粗く近似できることから，ある程度探索範囲を限定することが可能である．その限られた範囲において，各プロセスの実行時間の標準偏差が小さくなるような負荷分散を探索することにより，シミュレーションの総実行時間を調整前の約 55%まで短縮できることを実証した．

キーワード：高性能計算，静的負荷分散，ベイズ最適化

Automatic Load Balancing Using Bayesian Optimization for a Flood Simulation Code

Ayumu Ishizuka[1,a)]　Takeshi Yamashita[2,3,b)]　Ryusuke Egawa[3,4,c)]　Hiroyuki Takizawa[3,d)]
Tao Yamamoto[5,e)]　So Kazama[5,f)]

Received: October 20, 2020, Accepted: March 19, 2021

Abstract: To reduce the execution time of an MPI parallel application, it is necessary to balance the loads among MPI processes. In the case of our target application, flood simulation, the land area assigned to each MPI process affects the performance of vector processing. Thus, it is difficult to find the optimal load balance in advance. To make matters worse, each location might need a different computation based on the type of land usage. Conventionally, an expert programmer finds an appropriate load balance by hand in a try-and-error fashion, which needs expertise and is also labor-intensive. Since one trial execution of the simulation is time-consuming, simple search algorithms such as grid search and random search are not practical to automate the load balancing. This work, therefore, proposes an automatic static load balancing method using Bayesian optimization and parameter space pruning, while assuming the flood simulation as our target application. Bayesian optimization can efficiently search the maximum value of a given unknown function, and needs less trials than grid search and random search to perform load balancing. In addition, in the case of the flood simulation, as the load of each MPI process can roughly be estimated by the area of land regions assigned to the process, the parameter search space can be limited based on the estimation. Within the limited search space, the proposed method balances the load among MPI processes so as to minimize the standard deviation of their execution times. This paper demonstrates that the proposed approach can properly reduce the total execution time of the flood simulation to approximately 55% of its original execution time.

Keywords: high performance computing, static load balancing, Bayesian optimization

1. 緒論

近年，高性能計算技術は大きく発展し，同時に大規模数値シミュレーションの実行が可能となった．たとえば，山本らは日本全国を対象に，降水量データを用いた洪水シミュレーションを行い，全国の浸水深を算出している [1], [2]．洪水シミュレーションでは，予測された浸水深と全国の土地利用データに基づいて都道府県ごとに想定される被害額を求め，地球温暖化にともなう気候変動への対策を検証している．このように，大規模数値シミュレーションの実行により，実験が困難な事象を予測することが可能となるため，その需要は今後さらに高まると予想される．

大規模数値シミュレーションは，多くの場合 MPI（Message Passing Interface）を用いた並列分散処理を行う [3]．しかし，大規模シミュレーションはメモリ律速である場合が多く，通常のプロセッサでは並列化を行ったうえでも多大な実行時間を要する．したがって，NEC SX-Aurora TSUBASA（SX-AT）に代表される，高いメモリバンド幅を持つプロセッサの利用が効果的である [4]．しかし，SX-AT はベクトルプロセッサであり，その性能はベクトル長に大きく依存する．このため，通常のプロセッサとは異なる負荷分散調整がしばしば求められる．

上述の洪水シミュレーションの場合，事前に静的な調整を行うため，従来は熟練のプログラマが手作業で負荷の均等化を行ってきた．しかし，担当プログラマがシミュレーションの計算内容に精通しているとは限らず，むしろ計算内容をブラックボックスとして並列化作業を行う事例も多いため，その作業には経験と試行錯誤に基づく多大な労力が必要である．ブラックボックス関数の最適化手法として，グリッドサーチやランダムサーチがあげられる．調整可能なパラメータが複数存在する場合，グリッドサーチよりもランダムサーチの方が高効率である [5]．しかし，本研究で扱う洪水シミュレーションでは 1 回試行するごとに無視できない実行時間がかかるため，現実的な時間内に最適化を行うためにはより適応的な探索手法が必要である．他の負荷分散手法には，負荷の最も小さいプロセッサに対して実行時に動的にタスクを割り当てる方法や，事前に定めた閾値の範囲内で順にタスクを割り当てる方法が存在する [6], [7]．しかし，洪水シミュレーションでは，プロセス間で頻繁にデータ転送を行い同期をとる必要がある．したがって，このような柔軟にプロセスを割り当てる手法を適用することは困難である．さらに，シミュレーション規模の増大にともない，並列化時の分割方法は指数関数的に増加するため，これらの探索方法では今後対応できなくなることが強く懸念される．また，N 体計算や MPS 法に代表される粒子系シミュレーションでは，動的負荷分散による高速化が報告されているが [8], [9]，洪水シミュレーションは粒子法ではなく格子法による計算を行うため，実行時に動的に負荷変動が生じることはない．そのため，本研究では静的負荷分散のみを対象とする．このように，シミュレーションの実行時間を短縮するため，並列化時の負荷分散調整を自動化する効果的な方法が求められている．

本研究では，前述の洪水シミュレーションを SX-AT に移植して高速化を図るとともに，ベイズ最適化 [10] と探索範囲の調整による静的負荷分散の自動化を提案する．ベイズ最適化とは，ベイズ推定に基づいて未知の関数の最大値，あるいは最小値を効率的に探索する方法であり，これを用いることでグリッドサーチやランダムサーチに比べて少ない試行回数で負荷分散が可能となる．また，須田はソフトウェア自動チューニングにベイズ統計に基づく手法を取り入れており，コストが不確かな状況ではベイズ的手法は効果的であることを報告している [11]．しかし，並列計算の分割方法は膨大な数に及ぶ．前述の洪水シミュレーションでは，全国を南北方向に 250 m 間隔でメッシュ分割しており，その格子点数は 8,958 に及ぶ．8,958 格子点数を 32 プロセスに分割する際，その組合せは式 (1) で求められる．すなわち，格子点数を 1 つ除いた 8,957 格子点数から，分割点となる $32-1=31$ 格子点を選ぶ組合せの問題として求めることができる．

$$ {}_{8957}\mathrm{C}_{31} = \frac{8957!}{8926!31!} \tag{1} $$

式 (1) より，本シミュレーションの分割方法は少なくとも $(8927/31)^{31}$ 通り以上存在することが分かる．これより，すべての分割方法を探索候補とした場合，安直にベイズ最適化を用いるだけでは最適解が得られる確率は著しく低下する．ベイズ最適化における高次元最適化や目的関数の制約条件に関する研究は数多く報告されているが [12]，洪水シミュレーションの領域分割における組合せ最適化のように，説明変数に制約条件を付けつつ，高次元最適化である静的負荷分散にベイズ最適化を行った事例は著者らの知る限りでは報告されていない．したがって，明らかに最適解が得られないような探索候補を除外し，探索範囲を効果的

1 東北大学大学院情報科学研究科
Graduate School of Information Sciences, Tohoku University, Sendai, Miyagi 980–8578, Japan
2 東北大学情報部情報基盤課
Information Infrastructure Division, Information Department, Tohoku University, Sendai, Miyagi 980–8578, Japan
3 東北大学サイバーサイエンスセンター
Cyber Sciencecenter, Tohoku University, Sendai, Miyagi 980–8578, Japan
4 東京電機大学
Tokyo Denki University, Adachi, Tokyo 120–8551, Japan
5 東北大学大学院工学研究科
Graduate School of Engineering, Tohoku University, Sendai, Miyagi 980–8579, Japan
a) ishizuka@hpc.is.tohoku.ac.jp
b) yamacta@tohoku.ac.jp
c) egawa@mail.dendai.ac.jp
d) takizawa@tohoku.ac.jp
e) tao.yamamoto.r4@dc.tohoku.ac.jp
f) so.kazama.d3@tohoku.ac.jp

に限定する必要がある．

本論文では，洪水シミュレーションの陸地領域に対する計算内容をブラックボックスとして扱い，その適切な静的負荷分散を実現する手法を提案する．つまり，プログラムの詳細を精査せずに，発見的手法を用いて静的負荷分散を行う．これは，多くの場合シミュレーション開発者と高速化支援者が異なり，高速化支援者が計算内容の詳細を完全に把握することは困難なためである．また，とりうる領域分割の組合せ数は膨大となるが，各プロセスが担当する陸地領域の面積から負荷を粗く近似できることから，ある程度探索範囲を限定することができる．その限られた範囲において，各プロセスの実行時間の標準偏差が小さくなるような負荷分散を探索することにより，シミュレーションの総実行時間を短縮できることを実証する．

2. ベイズ最適化

2.1 ベイズ最適化の組合せ最適化問題への適用

ベイズ最適化 [10] とは，未知の目的関数 $f(\boldsymbol{x})$ を確率変数と見なし，ベイズ推定に基づいて $f(\boldsymbol{x})$ が最大値，あるいは最小値をとるような入力 $\boldsymbol{x}$ を効率的に探索する手法である．ベイズ最適化は，機械学習におけるハイパーパラメータ調整のように，手がかりの少ない入出力関係を最適化する有力な方法である [14]．したがって，本研究で扱う洪水シミュレーションの並列化に関するパラメータと実行時間の関係についても，同様にベイズ最適化を用いることが効果的であると考えられる．しかし，シミュレーションの最適な分割方法を求める問題は組合せ最適化問題であるため，入力 $\boldsymbol{x}$ には制約条件が存在する．よって，効率的にベイズ最適化を行うためには入力 $\boldsymbol{x}$ の扱い方を工夫する必要がある．

組合せ最適化問題に適用される典型的なアルゴリズムとして，遺伝的アルゴリズムがあげられる [15]．遺伝的アルゴリズムでは，解の候補を数値の並びで表現したものを遺伝子と見なし，遺伝子に対応した個体の評価を基に遺伝子集合を逐次的に更新する．遺伝的アルゴリズムの長所として，遺伝子の大きさに関する制約が少ないことから，大規模な問題に対して適用可能であることがあげられる．一方で，単に優良個体を選択していくだけでは局所最適解に陥る可能性が高いという欠点が存在する．このため，遺伝子集合の更新には優良個体の選択だけでなく，遺伝子の組み換えや突然変異等，様々な状況を設定し，その確率を適切な値に調整する必要がある．これには，直感や経験に頼った様々な工夫が必要とされる．

遺伝的アルゴリズムとは異なり，ベイズ最適化では確率的な予測に基づいた探索が行われるため，局所解に陥ることなく大域的な最適化が可能である．また，シミュレーションの負荷分散を最適化するという問題の特性上，適切な分割方法はある程度予測可能である．これより，探索範囲を適切に限定することで，制約条件を含む入力 $\boldsymbol{x}$ の有望な候補をより効率的に選択することが可能となる．このため，本研究ではベイズ最適化を用いて静的負荷分散を行うことを考える．ベイズ最適化はガウス過程回帰法，および獲得関数の 2 つの概念から成り立っているため，2.2 節および 2.3 節でそれぞれ説明する．

2.2 ガウス過程回帰法

ガウス過程回帰法とは，ガウス過程に基づいて回帰分析を行う手法である [10]．本節では，はじめにガウス過程について説明し，次にガウス過程に基づく回帰分析について述べる．

2.2.1 ガウス過程

ガウス過程では，いかなる N 個の入力の集合 $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)$ についても，対応する出力 $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ の同時分布 $p(\boldsymbol{y})$ が多変量ガウス分布に従うと定義する．ただし，N 次元のベクトル $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ が平均 $\boldsymbol{\mu}$，共分散行列 $\boldsymbol{K}$ の多変量ガウス分布 $\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{K})$ に従うとき，確率密度関数は式 (2) で表される．

$$\begin{aligned}&\mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}, \boldsymbol{K})\\ &= \frac{1}{(\sqrt{2\pi})^N \sqrt{|\boldsymbol{K}|}} \exp\left(-\frac{1}{2}(\boldsymbol{y}-\boldsymbol{\mu})^T \boldsymbol{K}^{-1}(\boldsymbol{y}-\boldsymbol{\mu})\right)\end{aligned} \tag{2}$$

ここで，共分散行列 $\boldsymbol{K}$ は N 次正方行列であり，その要素 $K_{nn'}$ は y_n と $y_{n'}$ の共分散を表している．すなわち，$K_{nn'}$ は式 (3) で表される．

$$\begin{aligned}K_{nn'} &= \mathbb{E}[(y_n - \mathbb{E}[y_n])(y_{n'} - \mathbb{E}[y_{n'}])]\\ &= \mathbb{E}[y_n y_{n'}] - \mathbb{E}[y_n]\mathbb{E}[y_{n'}]\end{aligned} \tag{3}$$

すべての y_n と $y_{n'}$ の組合せは $\boldsymbol{y}\boldsymbol{y}^T$ で表されることから，$\boldsymbol{K}$ は式 (4) のように表される．

$$\boldsymbol{K} = \mathbb{E}[\boldsymbol{y}\boldsymbol{y}^T] - \mathbb{E}[\boldsymbol{y}]\mathbb{E}[\boldsymbol{y}]^T \tag{4}$$

ここで，出力 y を入力 $\boldsymbol{x}$ の何らかの関数である $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \ldots, \phi_M(\boldsymbol{x}))$ と重み $\boldsymbol{w} = (w_1, w_2, \ldots, w_M)^T$ を用いて表すと，式 (5) のとおりになる．

$$\boldsymbol{y} = \boldsymbol{\Phi}\boldsymbol{w} \tag{5}$$

このとき，式 (4) より

$$\begin{aligned}\boldsymbol{K} &= \mathbb{E}[\boldsymbol{y}\boldsymbol{y}^T] - \mathbb{E}[\boldsymbol{y}]\mathbb{E}[\boldsymbol{y}]^T = \mathbb{E}[(\boldsymbol{\Phi}\boldsymbol{w})(\boldsymbol{\Phi}\boldsymbol{w})^T]\\ &= \boldsymbol{\Phi}\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T]\boldsymbol{\Phi}^T = \lambda^2 \boldsymbol{\Phi}\boldsymbol{\Phi}^T\end{aligned} \tag{6}$$

となる．ここで，$\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^T] = \lambda^2 \boldsymbol{I}$ とした．式 (6) より，$\boldsymbol{K}$ の要素 $K_{nn'}$ は式 (7) で表される．

$$K_{nn'} = \lambda^2 \boldsymbol{\phi}(\boldsymbol{x}_n)\boldsymbol{\phi}(\boldsymbol{x}_{n'})^T \tag{7}$$

ガウス過程によると，$\boldsymbol{y}$ の分布は共分散行列 $\boldsymbol{K}$ の要素，すなわち式 (7) のみを用いて表すことができる．この値は $\boldsymbol{x}_n$ と $\boldsymbol{x}_{n'}$ の類似度を数値化したものであるため，$\boldsymbol{\phi}(\boldsymbol{x}_n)$ や $\boldsymbol{\phi}(\boldsymbol{x}_{n'})$ を直接計算する必要はなく，新たに $K_{nn'}$ の値を与える関数を定義すれば十分である [10]．これを $\boldsymbol{x}_n$ と $\boldsymbol{x}_{n'}$ のカーネル関数（kernel function）と呼び，式 (8) で定義する．

$$K_{nn'} = k(\boldsymbol{x}_n, \boldsymbol{x}_{n'}) \tag{8}$$

動径基底関数（radial basis function，RBF）カーネルは，入力 $\boldsymbol{x}$ と $\boldsymbol{x}'$ の類似度を表すために最もよく使われるカーネル関数であり，式 (9) で定義される．ここで，$\theta_1, \theta_2 \in \mathbb{R}$ はこのカーネル関数の性質を決定するパラメータである．

$$k(\boldsymbol{x}, \boldsymbol{x}') = \theta_1 \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{x}'|^2}{\theta_2}\right) \tag{9}$$

2.2.2　ガウス過程に基づく回帰分析

次に，式 (10) に示す N 個の入出力結果が与えられたとき，ガウス過程に基づき回帰分析を行う方法について述べる．

$$\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_N, y_N)\} \tag{10}$$

簡単のため，y は平均が 0 となるように正規化されているものとする．さらに，$\boldsymbol{x}$ と y の間に $y = f(\boldsymbol{x})$ の関係があり，この関数 f は平均 $\boldsymbol{0}$ のガウス過程から生成されているものとする．ここで，$\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ とおけば $\boldsymbol{y}$ はガウス分布に従い，共分散行列 $\boldsymbol{K}$ を用いて，

$$\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}) \tag{11}$$

と表現される．このとき，未知の入力 $\boldsymbol{x}^*$ に対する出力 y^* の値は式 (12) で表される．

$$\begin{pmatrix} \boldsymbol{y} \\ y^* \end{pmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{pmatrix} \boldsymbol{K} & \boldsymbol{k}_* \\ \boldsymbol{k}_*^T & k_{**} \end{pmatrix}\right) \tag{12}$$

ただし，

$$\boldsymbol{k}_* = (k(\boldsymbol{x}^*, \boldsymbol{x}_1), k(\boldsymbol{x}^*, \boldsymbol{x}_2), \ldots, k(\boldsymbol{x}^*, \boldsymbol{x}_N))^T \tag{13}$$

$$k_{**} = k(\boldsymbol{x}^*, \boldsymbol{x}^*) \tag{14}$$

は未知の入力 $\boldsymbol{x}^*$ と既知の入力の類似度をカーネル関数で求めベクトル化したもの，および $\boldsymbol{x}^*$ どうしの類似度である．式 (12) は $\boldsymbol{y}$ と y^* の同時分布であるため，$\boldsymbol{y}$ が与えられたときの y^* の条件付き確率は，多変量ガウス分布の条件付き確率として求めることができる．式 (15) がガウス過程回帰法で実際に求められる予測分布である．

$$p(y^*|\boldsymbol{x}^*, \mathcal{D}) = \mathcal{N}(\boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{y}, k_{**} - \boldsymbol{k}_*^T \boldsymbol{K}^{-1} \boldsymbol{k}_*) \tag{15}$$

2.3　獲得関数

獲得関数とは，ガウス過程回帰法により求められた目的関数 $f(\boldsymbol{x})$ の予測分布を基に，次の探索候補点を決定する評価関数である [10]．具体的には，探索範囲内の各点における事後確率の期待値 $\mu(\boldsymbol{x})$ と標準偏差 $\sigma(\boldsymbol{x})$ を用いて獲得関数を計算し，その値が最大となる点を次の探索候補とする．真の最適解は，$\mu(\boldsymbol{x})$ が大きな値をとる暫定最適解の周辺に存在する可能性が高いが，周辺が局所最適解となっている可能性も考慮する必要がある．すなわち，$\sigma(\boldsymbol{x})$ が大きい値をとるような，探索が進んでいない領域についても探索を行う必要がある．本研究で用いる獲得関数 $a_{EI}(\mu, \sigma)$ は期待改善度（expected improvement）と呼ばれ，式 (16) で定義される．

$$\begin{aligned} a_{EI}(\mu, \sigma) &= \mathbb{E}[(f - \tau)I(f > \tau)] \\ &= (\mu - \tau)\Phi(t) + \sigma\phi(t) \end{aligned} \tag{16}$$

ここで，τ は事前に行われた N 回の観測 $y_N = f(\boldsymbol{x}_N)$ の中で最大の y を表す．$t = (\mu - \tau)/\sigma$ は，各 $\boldsymbol{x}$ における μ と τ の差を標準偏差で正規化したものである．$I(f > \tau)$ は，$f > \tau$ が成り立つとき 1，成り立たないとき 0 の値をとる指標関数である．$\Phi()$，$\phi()$ は，それぞれ標準正規分布の累積分布関数と密度関数である．

期待改善度による探索を時系列に沿って可視化したものを**図 1** (a)，図 1 (b)，および図 1 (c) に示す．各図のオレンジのグラフは目的関数の真値を表し，緑の点は実測値，青線および水色の領域はガウス過程回帰法により得られた目的関数の期待値と標準偏差を表す．図 1 (a) より，目的関数の期待値は探索範囲の中央付近で最大値をとることが確認できる．しかし，図 1 (b) を見ると，新たに観測を行ったのは図の右側の領域である．したがって，期待値だけでなく，探索の進んでいない領域も考慮して探索候補点を決定していることが分かる．そして，図 1 (b) の時点で右側の領域に最大値が存在する確率が低いことが明らかになったため，図 1 (c) では探索範囲の中央付近で新たに観測を行い，最大値に近い値を発見している．このように，獲得関数を用いた探索を行うことで，局所最適解に陥ることなく探索を続けることが可能となる．

3.　ベイズ最適化に基づく静的負荷分散手法

本論文では，洪水シミュレーションの静的負荷分散をベイズ最適化に基づく方法で実現する．ただし，とりうる領域分割の組合せ数は膨大となるため，ベイズ最適化を活かすためには探索範囲を適切に限定する必要がある．また，探索範囲を限定する場合，探索範囲内に最適解が存在することは保証できない．したがって，試行回数 N 回のベイズ最適化を 1 セットとし，試行結果に基づき探索範囲を更新する方法を提案する．なお，探索範囲の更新により過去の試行結果の多くは新たな探索範囲から外れてしまうた

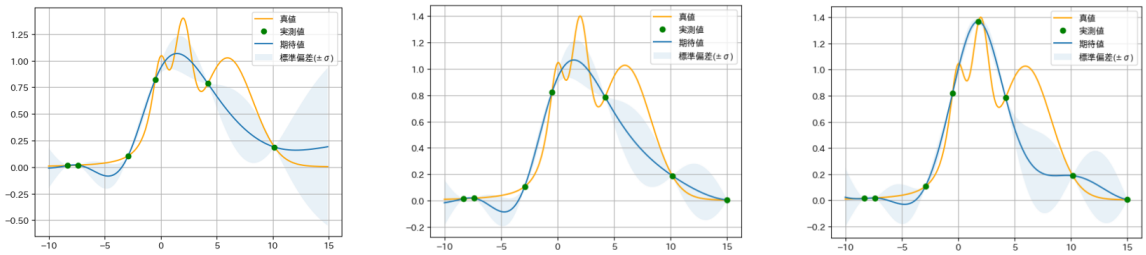

図 1　獲得関数による探索
Fig. 1　Search based on an acquisition function.

め，ベイズ最適化で過去の試行結果は引き継がないものとする．ベイズ最適化のセットを繰り返す手順は，以下のとおりである．

手順 1　初期探索範囲（1 セット目）を陸地均等分割の $\pm\alpha$%に設定
手順 2　ベイズ最適化（試行回数 N 回）
手順 3　上位の説明変数値の平均の $\pm\alpha$%を新たな探索範囲に設定
手順 4　手順 2 へ戻る（セット番号+=1）

以下，3.1 節では対象とする洪水シミュレーションの概要を述べる．3.2 節ではベイズ最適化を適用する際の制約条件について説明する．3.3 節では目的関数の設定方法について述べる．3.4 節では探索範囲の限定方法について説明する．

3.1　対象とする洪水シミュレーションの概要

高性能計算システムはさらなる高性能化のため，複雑化が進んでいる [13]．したがって，現在の高性能計算システムを最大限に活用可能なアプリケーションを記述することは，以前にも増して困難になっている．その一例として，洪水シミュレーションを高精度に行うことを考える．山本らの手法 [1] では，高精度化のために日本全国を 250 m 四方のメッシュで区切って計算を行っている．この場合，計算要素数は膨大な数に及ぶため，MPI による並列化が必要になる．

本研究で扱う洪水シミュレーションでは，**図 2** に示すとおり日本全国を南北方向に M 分割し，M プロセスの並列計算にすることで実行時間の短縮を実現している．ただし，浸水深の計算が行われるのは陸地領域のみであり，計算負荷が陸地領域に集中する．このことから，負荷を均等化するための安直な方法として，各プロセスが処理する陸地面積を等しくするような分割方法が考えられる．この分割方法を，**陸地均等分割法**と呼ぶことにする．

図 3 は，Intel Core i7-9700K を 4 基使用したクラスタで，陸地均等分割法によりシミュレーションを実行した際の各プロセスの実行時間を表す．なお，本研究ではプロセスごとの負荷を明確にするため，その計算時間は総実行時

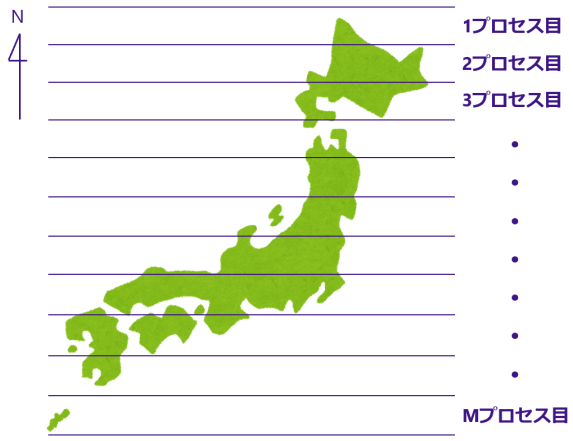

図 2　洪水シミュレーションにおける全国の分割方法
Fig. 2　Domain decomposition for flood simulation.

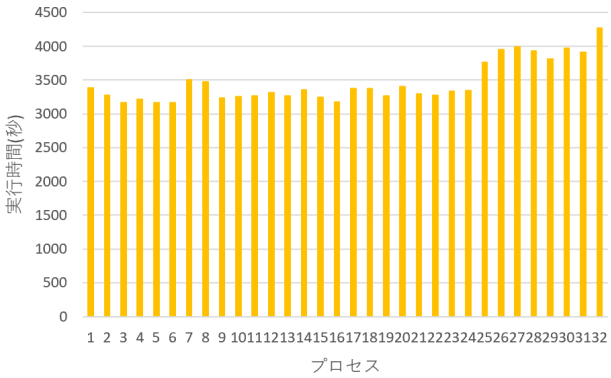

図 3　Intel Core i7-9700K における陸地均等分割時の実行時間
Fig. 3　The execution time on Intel Core i7-9700K when equally dividing land areas.

間からデータの入出力およびプロセス間通信に要した時間を除外して算出されている．MPI による並列数は 32 とし，1 つのコアが 1 プロセスを担当する形で並列化を行っている．また，ランク番号の若いプロセスが北側の陸地の計算を担当するように担当領域が割り当てられている．図 3 より，一般的なプロセッサを用いてシミュレーションを行った場合，実行時間の差は最大でも 1.3 倍程度であり，負荷を比較的均一化できていることを確認できる．一方で，シミュレーション実行時間は 4,000 秒を超えているため，よ

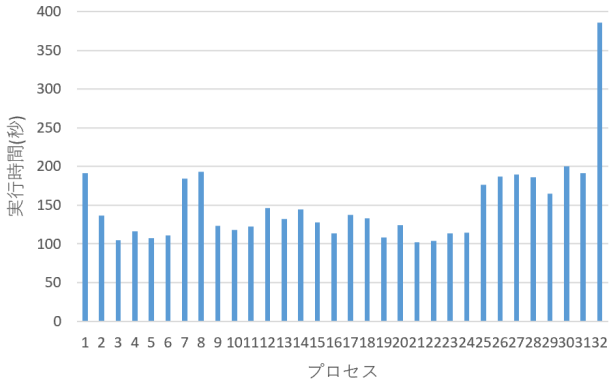

図 4　SX-AT における陸地均等分割時の実行時間
Fig. 4　The execution time of SX-AT when equally dividing land areas.

り多くの条件でシミュレーションを行うためにはその高速化が強く求められる．

図 4 は，東北大学サイバーサイエンスセンター内の NEC SX-Aurora TSUBASA（SX-AT）を用いたシステム[*1]上で，陸地均等分割法によりシミュレーションを実行した際の各プロセスの実行時間を表す．図 4 より，SX-AT を用いることで，実行時間を 10 倍以上短縮できていることが分かる．これは，洪水シミュレーションの計算カーネルがメモリ律速であり，SX-AT のベクトルエンジン（VE）を活用することでボトルネックが解消されたことを意味する．一方で，SX-AT 上では，陸地均等分割法により必ずしも最適な負荷分散が得られるわけではないことが分かる．特に，日本地図の最南端部分を担当する 32 番目のプロセスの実行時間が極端に増加している．これは，日本の最南端部分において陸地領域が点在していることから，図 2 の横方向に陸地領域の計算を行うループの長さが短くなり，SX-AT のベクトル演算性能が十分に活用されなかったためである．また，洪水による被害額を算出する過程では，計算要素ごとに土地利用データに基づいた計算を行う．具体的には，建物用地では家屋の抵抗が考慮され，さらに県別の土地の価値に基づいて被害額が計算される．したがって，これを考慮した負荷分散調整はさらに困難なものとなる．そのうえ，負荷の分布はシミュレーションの並列数や入力データに応じて変化するため，実行条件ごとに負荷分散が求められる．

以上より，シミュレーションの負荷分散を実現するには，陸地面積の割合だけでなくハードウェアの特性等も判断したうえで，各プロセスが担当する計算領域の南北方向の幅を決定する必要があることが分かる．以降，この南北方向の幅を演算範囲と呼ぶ．このため，本論文では各プロセスが担当する演算範囲を説明変数として扱い，洪水シミュレーションにベイズ最適化を適用する．

[*1] システムの仕様は 4.1 節で後述する．

3.2　制約条件

本研究では，各プロセスの演算範囲を説明変数として考え，ベイズ最適化を適用する．ただし，各プロセスの演算範囲について探索を行う際に，各プロセスの演算範囲の総和が全国の南北方向の格子点数に一致しなければならないという制約条件がある．すなわち，M プロセスの並列計算を行う際の各プロセスの演算範囲を $x_1, x_2, \ldots, x_M \in \mathbb{N}$，全国の南北方向の格子点数を $X \in \mathbb{N}$ とすると式 (17) が成立しなければならない．

$$\sum_{i=1}^{M} x_i = X \tag{17}$$

ベイズ最適化を実行する際，初期観測点となる説明変数の組合せはランダムに決定されるため，その総和が一定となるような制約を加えることは困難である．したがって，探索を行う説明変数を 1 つ削減することで，制約条件の緩和を試みる．具体的には，探索を行う $M-1$ プロセスの演算範囲を $x_1, x_2, \ldots, x_{M-1} \in \mathbb{N}$ とすると，制約条件は式 (18) に置き換えられる．

$$\sum_{i=1}^{M-1} x_i < X \tag{18}$$

このように，制約条件を等式である式 (17) から不等式である式 (18) へ変形することで，各説明変数の値がランダムに決定される際に制約条件が満たされる確率を十分に高くすることができる．このとき，残りの 1 プロセスの演算範囲 x_M は式 (19) で与えられる．

$$x_M = X - \sum_{i=1}^{M-1} x_i \tag{19}$$

ただし，制約条件である式 (18) が満たされない場合，このような分割方法は存在せず，シミュレーションを実行することは不可能である．したがって，制約条件が満たされなかった場合，目的関数の出力を十分に大きな値に設定する必要がある．

3.3　目的関数

目的関数とは，2.3 節で示したとおり，ベイズ最適化で探索を行う未知の関数のことである．本研究では，洪水シミュレーションの実行時間を短縮することが最終的な目標である．したがって，最も単純な方法として，シミュレーションの実行時間，すなわち各プロセスの実行時間の内最大の値を目的関数の出力 y として設定する方法が考えられる．実際に，全体の実行時間に影響を及ぼすプロセス数は限られているため，単に実行時間を評価するのみである程度の実行時間短縮は可能である．しかし，実行時間そのものを評価する場合，目的関数の最小値に到達する前に局所解に陥ってしまうことが懸念される．これは，3.2 節で述

べたとおり，説明変数の総和が一定となるような制約条件が存在することに起因する．すなわち，実行時間を短縮させるためにあるプロセスの演算範囲を減少させると，必ず他のプロセスの演算範囲が増加し，そのプロセスの実行時間が増加する．これにより，ある程度最適化が進んだ際に，説明変数の調整が実行時間の比較的長い数プロセスに限定され，実行時間が短く十分に活用されていないプロセスが残されたまま探索が終了する恐れがある．したがって，単にシミュレーションの実行時間を評価するだけでは不十分であり，より高い確率で最適解に近づくことができる目的関数が求められる．

本研究では，シミュレーションの実行時間ではなく，各プロセスの実行時間の標準偏差を目的関数の出力 y とし，これを最小化する．すなわち，各プロセスの演算範囲を $x_1, x_2, \ldots, x_{M-1} \in \mathbb{N}$，各プロセスの実行時間を $t_1, t_2, \ldots, t_M \in \mathbb{R}$ としたとき，目的関数は式 (20) で表される．

$$y = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(t_i - \bar{t})^2} = f(x_1, x_2, \ldots, x_{M-1}) \qquad (20)$$

ここで，$\bar{t}$ は各プロセスの実行時間の平均値である．実行時間が $\bar{t}$ と大きく異なるプロセスが増加したとき，すなわち負荷分散が悪化した場合に目的関数の出力 y は増加し，負荷分散が改善された場合 y は減少する．y の最小値は 0 である．

標準偏差を最小化する場合，標準偏差の減少に従ってシミュレーション実行時間も減少するとは限らないため，短期的には実行時間を直接評価した方が効率的である可能性はある．しかし，ある程度最適化が進むと，過去の試行結果から全体の負荷分散に大きく影響しているプロセスを特定することができるため，最終的にシミュレーション実行時間も短縮されることが期待される．また，負荷分散が悪化したとき，標準偏差の増加率はシミュレーション実行時間の増加率に比べてゆるやかになるため，例外処理の設定も容易になる．すなわち，探索候補となる説明変数が制約条件を満たさない場合や，シミュレーション実行時間が一定の値を超過し強制終了されたときに，ある程度大きな値を出力 y として設定することで，このような説明変数の値が不適切であることを学習させることができる．

このように，目的関数に式 (20) を用いることで，シミュレーションの実行時間が最小となるような説明変数 $x_1, x_2, \ldots, x_M \in \mathbb{N}$ を効率的に探索することができる．

3.4　探索範囲の限定方法

ベイズ最適化を実行する際には，説明変数の探索範囲を適切に設定する必要がある．3.1 節で述べたとおり，洪水シミュレーションでは陸地領域のみで浸水深の計算が行われるため，各プロセスが扱う陸地面積の大きさが負荷分散に大きく影響する．これより，陸地均等分割法を行った際の各プロセスの演算範囲に対し，$\pm\alpha$%を各説明変数の初期探索範囲として活用する．ここで，α を必要以上に大きな値にすると，現実的な試行回数以内に探索範囲全体を調べることが困難になり，最適化が進まなくなる．経験的に，高次元最適化において目的関数が収束するには少なくとも数百回程度の試行が必要である [16], [17]．したがって，α は数百回程度の試行回数で最適化が可能な範囲内の値とする．ただし，α を十分大きな値に設定することが不可能であることから，初期探索範囲内に最適解が存在することは保証できない．

本提案手法では，探索結果に基づく探索範囲の更新を考える．すなわち，初期探索範囲内で行われた N 回の試行で，最も負荷分散に優れた説明変数の組をいくつか選択し，その平均値 $\pm\alpha$%を新たな探索範囲として設定する．3.3 節で述べたとおり，負荷分散を改善することで極端に実行時間の長いプロセスが特定され，実行時間が短縮される確率は高まることから，探索範囲の更新により局所解に陥ることなく探索を継続することが可能となる．

以上より，提案手法では初期探索範囲の中央値に陸地均等分割時の値を設定し，探索範囲の更新を複数回行うことで探索範囲を効果的に限定する．この探索範囲の限定方法の妥当性と効果に関しては，4 章で評価結果に基づいて議論する．

4.　性能評価と考察

4.1　評価条件

本洪水シミュレーションでは，東北大学サイバーサイエンスセンターに設置された NEC SX-Aurora TSUBASA A300-8 を使用する．本計算機は，アプリケーション演算処理を行うベクトルエンジン（VE）8 基と，OS 処理を主に行うベクトルホスト（VH）1 基から構成される．また，1 つの VE は 8 つのベクトルプロセッサコアで構成され，48 GB のメモリを共有している．したがって，本計算機を使用することで，最大で $8 \times 8 = 64$ 並列の計算を実行することが可能である．洪水シミュレーションは 32 プロセスで実行するように調整されたものであるため，本論文では VE を 4 基（すなわち 32 コア）使用して，32 並列計算の性能を評価する．SX-Aurora TSUBASA A300-8 の詳細を **表 1** に示す [18]．

本評価では，MPI により全国の南北方向の演算範囲が 32 分割され，32 プロセスの並列計算が行われる．したがって，提案手法でベイズ最適化が探索を行う説明変数は各プロセスが担当する演算範囲とし，説明変数の数は最南端を担当するプロセスを除き 31 とする．初期探索範囲を決定するパラメータは $\alpha = 20$ とする．1 セットの探索回数は $N = 200$ とし，1 セットごとに探索結果に基づく探索範囲の更新を行う．なお，本評価条件では 5 セットの探索を行

表 1　SX-Aurora TSUBASA A300-8 の詳細
Table 1　Specification of SX-Aurora TSUBASA A300-8.

ベクトルエンジン（VE）		ベクトルホスト（VH）	
搭載 VE 数	8	Xeon プロセッサ数	2
搭載 VE	Type 10B	Xeon プロセッサ	Intel XEON Gold 6126
最大総 VE 演算性能（TFLOPS）	17.20	メモリ構成	DDR4 DIMM x 12
最大総 VE メモリ帯域（TB/s）	9.60	OS	Red Hat Enterprise Linux 7.3

うことで手動最適解と同様の負荷分散が達成されるため，5 セット目までの結果に基づいて議論する．

なお，探索範囲を限定することで陸地均等分割法に近い分割方法を試行しているため，説明変数が制約条件を満たさない確率や，シミュレーション実行時間が一定の値を超過し強制終了される確率は低い．強制終了される時間は陸地均等分割時の実行時間からさらに余裕を持ち 600 秒としているが，万が一制約条件が満たされずシミュレーションが実行されなかった場合や強制終了された場合を考慮し，そのときの目的関数の値を経験的に $y = 100$ と設定する．

各評価において，ベイズ最適化の実装には Python プログラム用のライブラリである bayesian-optimization を使用する [19]．

4.2　事前知識に基づく初期探索範囲の調整

はじめに，探索範囲の調整結果を**図 5** および**図 6** に示す．図 5 において，灰色の棒グラフは初期探索範囲であり，陸地均等分割時の演算範囲 ±20%の領域を表す．また，オレンジのグラフは，プログラマが手動で負荷分散を最適化したときの各プロセスの演算範囲である．この手動で調整された負荷分散で実行時間が十分に短くなるため，本議論ではこれを最適な負荷分散と考える．なお，手動調整により実行時間は約 220 秒にまで短縮され，図 4 に示す陸地均等分割法から約 45%実行時間が削減されている．図 5 より，手動最適化後の各プロセスの演算範囲は過半数が初期探索範囲内に収まっていることが分かる．すなわち，陸地均等分割法の分布傾向は，最適な負荷分散における演算範囲の分布に近いといえる．したがって，初期探索範囲に陸地均等分割法を利用することで，探索範囲を事前に調整せずに一様分布を用いる場合に比べ，探索範囲を最適解の近傍のみに適切に限定できていることが分かる．一方で，演算範囲の大きい 1，8，31 プロセス目の初期探索範囲は手動最適解との差が大きく，最適解が含まれるように探索範囲を調整する必要があることも分かる．

図 6 は，5 セット目の探索範囲，すなわち 200 × 4 回の探索に基づき更新された各プロセスの探索範囲を示す．図 6 より，手動最適解との差が大きかった 1，8，31 プロセス目の探索範囲が手動最適解に近づいたことが分かる．実際に，1，8，31 プロセス目の探索範囲の中央値について，最適解との差は調整前でそれぞれ 177，146，324 格子点数であったのに対して，調整後は 105，60，148 格子点数に縮まっている．また，その他のプロセスの探索範囲も手動最適解の分布に合わせて移動しており，計 31 プロセスの最適解との誤差の平均値は 57 格子点数から 30 格子点数に改善されている．このことから，探索結果に基づき探索範囲を更新することで最適解に近づくことが可能であるといえる．

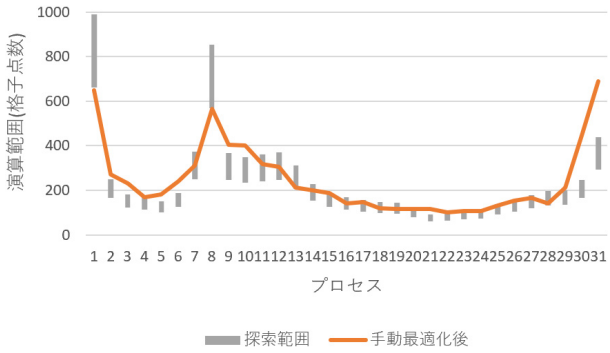

図 5　各プロセスの初期探索範囲，および手動最適化後の演算範囲
Fig. 5　Initial search range for each process, and manually-tuned optimal decomposition.

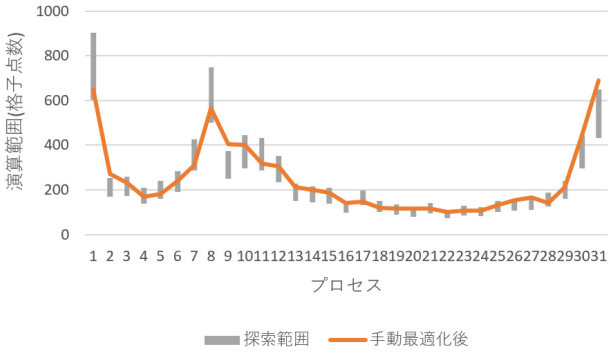

図 6　更新後の探索範囲，および手動最適化後の演算範囲
Fig. 6　Updated search range for each process, and manually-tuned optimal decomposition.

4.3　ベイズ最適化による静的負荷分散

次に，目的関数に式 (20) を用いた場合の実行結果を**図 7** に示す．横軸は 200 回 × 5 セットのベイズ最適化による合計の試行回数，縦軸は各試行回数までに得られた最小の標準偏差およびシミュレーション実行時間を表している．図 7 において，標準偏差の値が小さくなるほどシミュレーションの負荷分散は改善され，各プロセスの演算範囲は最適解に近づく．

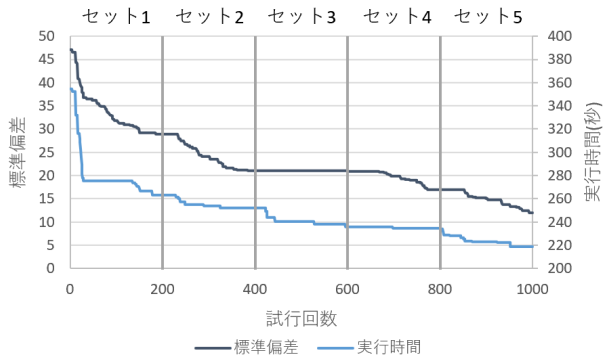

図 7　標準偏差に基づく最適化
Fig. 7　Optimization based on standard deviation.

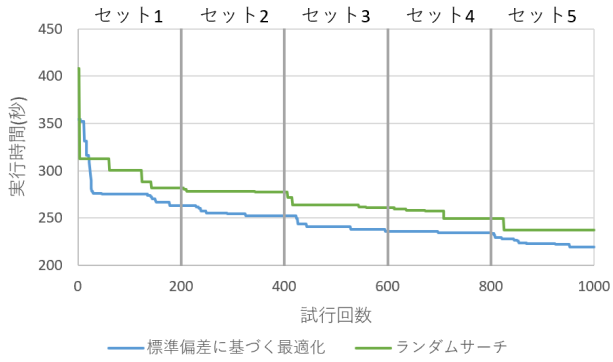

図 8　標準偏差に基づく最適化とランダムサーチの比較
Fig. 8　Comparison of the optimization based on standard deviation and random search.

図 7 より，1 セット目および 2 セット目の探索で標準偏差は大きく減少しているが，3 セット目で一度最適化が停滞していることが分かる．これは，4.2 節で示したとおり一部のプロセスの探索範囲が最適解から離れており，標準偏差を最小化する際のボトルネックとなっているためである．そして，4 セット目以降の探索，特に 700 回目以降の探索において標準偏差は再び減少し始め，負荷分散が大幅に改善されていることが分かる．これより，探索範囲を更新することでボトルネックとなるプロセスの探索範囲が改善され，最適化がよりよい方向に進んだことが確認できる．最終的に，各プロセスの実行時間の標準偏差が 12 となるまで負荷分散を改善することができた．

このように，標準偏差を評価することで局所解に陥ることなく探索を行うことができた．標準偏差が減少することでプロセス間の負荷の偏りが減少し，間接的にシミュレーション実行時間を削減することができる．図 7 より，計 1,000 回の探索において最小実行時間は減少し続けており，実行時間の面でも最適化が進んでいることが分かる．

図 7 のセット 2，3，4 に示されるとおり，各セットの 200 回の試行のうち，後半の試行では最小実行時間が更新されない傾向にあることも分かる．これは，探索範囲に最適解が存在しないプロセスが存在し，その実行時間が短縮されないためである．その結果として，総実行時間も短縮されなくなる．提案手法では，各セットの開始時に探索範囲が更新される．図 7 では次のセットで実行時間が短縮していることから，更新後の新たな探索範囲でより実行時間の短い解を発見できているといえる．このことから，提案手法は探索範囲を更新することで，効果的に解の範囲を限定できていることが示された．

図 7 から，2 セット目および 4 セット目に標準偏差が大幅に削減されていることが分かる．一方，シミュレーション実行時間が 5%以上も短縮されているのは，3 セット目および 5 セット目の開始直後である．シミュレーション実行時間には，最長の実行時間を持つプロセスの影響しか現れない．しかし，標準偏差を最小化することでプロセス間の実行時間の偏りを減らすことができる．このため，探索範囲が更新されて最長実行時間のプロセスがより良い解を発見した際に，シミュレーション実行時間が大幅に短縮されている．このことからも，標準偏差に基づいて負荷分散を行う場合に，探索範囲を更新することの重要性が示されている．

最終的に，提案手法で得られた最小のシミュレーション実行時間は約 220 秒であり，図 4 に示す陸地均等分割法から約 45%実行時間を削減することができた．さらに，この値は本評価で最適解として扱っている手動最適化の 220 秒に等しく，洪水シミュレーションの静的負荷分散を十分な精度で自動化できる点で本提案手法の有用性が示されたといえる．

4.4　ランダムサーチとの比較

本研究では，ベイズ最適化と探索範囲の限定方法を組み合わせることで並列シミュレーションの静的負荷分散を実現した．しかし，過去の探索結果から負荷分散に優れた解を選択し，探索範囲を更新する方法はランダムサーチ等の単純なアルゴリズムと組み合わせることも可能である．そこで，探索範囲の限定方法を適用した際のベイズ最適化とランダムサーチを比較し，ベイズ最適化の有用性について改めて評価する必要がある．

図 8 における緑のグラフは，標準偏差に基づく最適化と同様の探索範囲更新方法でランダムサーチを行った際の，各試行回数における最小実行時間を表す．図 8 の 2，3，5 セット目に示されるとおり，各セットにおいて 200 回探索を行っているにもかかわらず，最初の 20 回程度を除きほとんど実行時間が短縮されない傾向にあり，探索範囲内を効率的に調べることができていない．これにより，探索範囲の限定方法も十分に機能せず，各試行において最小実行時間はベイズ最適化につねに劣る結果となった．

また，**図 9** は標準偏差による最適化後，およびランダムサーチ後の各プロセスの実行時間を示す．図 9 より，ランダムサーチ後の最長実行時間は約 218 秒であり，その 1/2 以下の短時間で終了した負荷の小さいプロセスは 2，29，

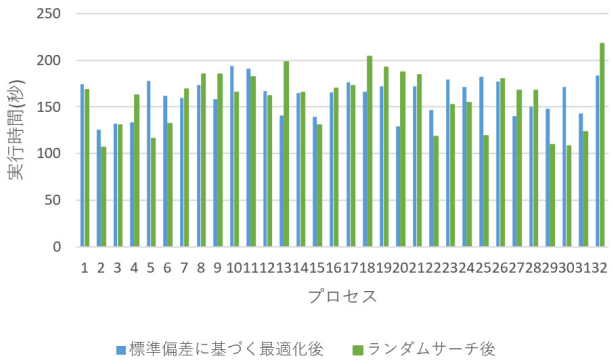

図 9　最適化後の各プロセスの実行時間
Fig. 9　The execution time of each process after optimization.

30 プロセス目の計 3 プロセスであった．一方で，標準偏差による最適化後の最長実行時間は約 194 秒であり，1/2 以下の短時間で終了したプロセスは存在しなかった．これより，ランダムサーチでは一部のプロセスを十分に活用できなかったのに対して，ベイズ最適化では全プロセスに負荷を分散させられたことが分かる．実際に，各プロセスの実行時間の標準偏差はランダムサーチ後で約 30，ベイズ最適化後で約 19 であり，ベイズ最適化の方が負荷分散に適した解を示した．

以上の結果から，探索範囲の限定方法を用いたうえでもランダムサーチによる静的負荷分散は困難であり，ベイズ最適化を用いる有用性が改めて確認できる．

5.　結論

本研究では，並列計算の一例である洪水シミュレーションを対象に，ベイズ最適化と探索範囲の限定方法を用いた静的負荷分散の自動化を提案した．

ベイズ最適化を並列計算の負荷分散に適用する際には，各プロセスが担当する演算範囲を説明変数とし，目的関数を各プロセスの実行時間の標準偏差とすることで局所解に陥ることなく探索が可能であることを示した．また，探索結果から優れた解を選択し，入力パラメータの平均値を新たな探索範囲の平均値とする探索範囲の限定方法を用いることで，効率的な探索が可能であることを示した．標準偏差に基づく評価と探索範囲の更新を組み合わせることで，シミュレーションの実行時間を陸地均等分割時の約 55%まで削減することができた．その結果，SX-AT に移植する前には 4,000 秒以上を要していた実行時間を約 220 秒にまで短縮し，同シミュレーションの飛躍的な高速化を達成することができた．この実行時間は，熟練者が手動で求めた最適解に等しく，実用上十分許容できる解が提案手法による自動調整によって得られたといえる．

今後の課題としては，手動調整がまだ行われていない 64 並列時の最適化や，シミュレーション実行中に負荷分散を評価し，演算範囲を更新する動的最適化を予定している．また，本研究の提案手法である探索範囲の限定方法には洪水シミュレーションの特徴を用いているため，他のシミュレーションに直接適用することは困難である．しかし，計算負荷の不均衡は他のシミュレーションにも起こりうるため，個々のシミュレーションに合わせた初期探索範囲の推定および調整は可能であると考えられる．そのような評価もまた，今後の課題としてあげられる．

謝辞　本研究は，文部科学省「次世代領域研究開発」量子アニーリングアシスト型次世代スーパーコンピューティング基盤の開発，JSPS 科研費 16H02822，および学際大規模情報基盤共同利用・共同研究拠点（課題番号：jh190014-NAH）の支援による．

参考文献

[1] 山本　道，風間　聡，峠　嘉哉，多田　毅，山下　毅：気候変動による洪水被害に対する緩和策と適応策の評価，地球環境研究論文集：地球環境シンポジウム/土木学会地球環境委員会（編），Vol.27, pp.15–23 (2019).
[2] 田中裕夏子，風間　聡，多田　毅，山下　毅，小森大輔：治水安全度を考慮した洪水・高潮リスク評価，水工学論文集 Annual journal of Hydraulic Engineering，JSCE/土木学会水工学委員会（編），Vol.64, pp.109–114 (2019).
[3] Gropp, W., Lusk, E., Doss, N. and Skjellum, A.: A high-performance, portable implementation of the MPI message passing interface standard, *Parallel Computing*, Vol.22, pp.789–828 (1996).
[4] Komatsu, K., Momose, S., Isobe, Y., Watanabe, O., Musa, A., Yokokawa, M., Aoyama, T., Sato, M. and Kobayashi, H.: Performance Evaluation of a Vector Supercomputer SX-Aurora TSUBASA, *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp.685–696 (2018).
[5] Bergstra, J. and Bengio, Y.: Random search for hyper-parameter optimization, *Journal of Machine Learning Research*, Vol.13, No.1, pp.281–305 (2012).
[6] Sharma, S., Singh, S. and Sharma, M.: Performance Analysis of Load Balancing Algorithms, *World Academy of Science, Engineering and Technology*, Vol.14 (2008).
[7] Leland, R. and Hendrickson, B.: An Empirical Study of Static Load Balancing Algorithms, *Proc. IEEE Scalable High Performance Computing Conference*, pp.682–685 (1994).
[8] Murotani, K., Koshizuka, S., Tamai, T., Shibata, K., Mitsume, N., Yoshimura, S., Tanaka, S., Hasegawa, K., Nagai, E. and Fujisawa, T.: Development of Hierarchical Domain Decomposition Explicit MPS Method and Application to Large-scale Tsunami Analysis with Floating Objects, *Journal of Advanced Simulation in Science and Engineering*, Vol.1, No.1, pp.16–35 (2014).
[9] Murotani, K., Koshizuka, S., Onigo, M., Shioya, R. and Nakabayashi, Y.: Development of Distributed Parallel Explicit Moving Particle Simulation (MPS) Method and Zoom Up Tsunami Analysis on Urban Areas, *The International Conference for High Performance Computing, Networking, Storage and Analysis*, No.11 (2014).
[10] 持橋大地，大羽成征：機械学習プロフェッショナルシリーズ ガウス過程と機械学習，講談社 (2019).
[11] 須田礼仁：オフライン自動チューニングの数理手法，IPSJ SIG Technical Report，Vol.2010-HPC-125, No.3 (2010).
[12] Greenhill, S., Rana, S., Gupta, S., Vellanki, P. and Venkatesh, S.: Bayesian Optimization for Adaptive Ex-

perimental Design: A Review, *IEEE Access*, Vol.8, pp.13937–13948 (2020).
[13] TOP500 Supercomputer Sites, available from ⟨https://www.top500.org⟩.
[14] 尾崎嘉彦，野村将寛，大西正輝：機械学習におけるハイパーパラメータ最適化手法：概要と特徴，電子情報通信学会論文誌，Vol.J103-D, No.9, pp.615–631 (2020).
[15] 篠崎隆宏，渡部晋治：音声認識とブラックボックス最適化，日本音響学会誌，Vol.72, No.10, pp.644–652 (2016).
[16] Rana, S., Li, C., Gupta, S., Nguyen, V. and Venkatesh, S.: High Dimensional Bayesian Optimization with Elastic Gaussian Process, *Proc. 34th International Conference on Machine Learning*, PMLR, Vol.70, pp.2883–2891 (2017).
[17] Sun, C., Jin, Y., Cheng, R., Ding, J. and Zeng, J.: Surrogate-Assisted Cooperative Swarm Optimization of High-Dimensional Expensive Problems, *IEEE Trans. Evolutionary Computation*, Vol.21, No.4, pp.644–660 (2017).
[18] NEC SX-Aurora TSUBASA A300-8, available from ⟨https://www.nec.com/en/global/solutions/hpc/sx/A300-8.html⟩.
[19] Fernando Nogueira, A Python implementation of global optimization with Gaussian processes, available from ⟨https://github.com/fmfn/BayesianOptimization⟩.

石塚 歩

2020 年東北大学工学部機械知能・航空工学科卒業．現在，同大学大学院情報科学研究科情報基礎科学専攻修士課程在籍．

山下 毅

2002 年東北大学大学院工学研究科博士課程前期修了．修士（工学）．2005 年同大学電気通信研究所に技術職員として勤務．2010 年同大学情報部情報基盤課勤務．2012 年より技術専門職員として高性能計算機に関する高速化を主とする利用者支援および管理・運用業務に従事．

江川 隆輔　（正会員）

2004 年東北大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．同年東北大学大学院情報科学研究科研究員，2005 年同助手．2007 年東北大学サイバーサイエンスセンター・助教，2013 年同准教授，大規模科学計算システムの運用・開発，コンピュータアーキテクチャ，高性能計算に関する研究に従事．2020 年東京電機大学工学部情報通信工学科教授，現在に至る．IEEE，電子情報通信学会各会員．

滝沢 寛之　（正会員）

1999 年 9 月東北大学大学院情報科学研究科博士後期課程修了．博士（情報科学）．同年 10 月新潟大学総合情報処理センター助手，2003 年 3 月東北大学情報シナジーセンター助手，2004 年 4 月同大学大学院情報科学研究科講師，2009 年 1 月同研究科准教授を経て，2017 年 1 月より東北大学サイバーサイエンスセンター教授，2019 年 4 月より同センターの副センター長も併任し，現在に至る．高性能計算の研究分野に広く興味を持ち，特に高性能計算システムやその性能を引き出すための基盤ソフトウェア技術の研究に従事している．また，東北大学サイバーサイエンスセンターにおいて，スーパーコンピュータ AOBA の設計および運用を主導している．電子情報通信学会，IEEE，ACM CS 各会員．

山本 道

2021 年東北大学大学院工学研究科修士課程修了．気候変動にともなう洪水激甚化に対する適応策評価についての研究に従事．同年日本工営株式会社入社．海外の水資源エネルギー開発事業に従事．

風間 聡

1990 年東北大学工学部土木工学科卒業．1995 年東北大学大学院工学研究科博士課程後期修了．博士（工学）．筑波大学構造工学系講師，タイ国アジア工科大学院講師，東北大学大学院環境科学研究科助教授を経て現職．環境研究総合推進費 S-18「気候変動影響予測・適応評価の総合的研究」に参加．土木学会，水文・水資源学会，国際水文科学会各会員．