様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

令和　２ 年　 ６ 月　 １ 日現在

機関番号：１４４０１
研究種目：基盤研究(A)（一般）
研究期間：2015～2019
課題番号：１５Ｈ０１６８７

研究課題名（和文）大規模グラフで表現された不規則・複雑な対象を高速にシミュレーションする方法の研究

研究課題名（英文）Study on high-speed simulation of irregular and complex objects represented by large-scale graph

研究代表者
萩原　兼一（Hagihara, Kenichi）

大阪大学・情報科学研究科・招へい教授

研究者番号：００１３３１４０

交付決定額（研究期間全体）：（直接経費）　　33,400,000円

研究成果の概要（和文）：大規模かつ不規則なグラフとしてシミュレーションの仕様が与えられ，そのシミュレーションを高速に行う並列プログラムを効率良く自動生成する技術を確立した．具体的には，まずプログラム生成の高速化に関して，グラフを粗粒化してから分割することで自動生成の並列化率を向上し，細粒グラフ分割と比較して約10倍の高速化を達成した．次に生成するプログラムに関して，GPU，CPUクラスタ，およびベクトル型スーパーコンピュータの3つ並列計算機に応じた最適化を施し，最大約2倍の速度向上を実現した．
本研究の主な応用例である生体シミュレータを用いると，ユーザはプログラムを作成することなく多様な並列計算機の恩恵を得られる．

研究成果の学術的意義や社会的意義
大規模で不規則かつ複雑な対象を高速にシミュレーションすることが可能となり、モデル作成も含めたシミュレーション過程全体の時間を短縮できる。したがって、これらのシミュレーションを必要とする多くの科学分野の研究を加速する。また、科学シミュレーションに関する貢献だけでなく、今後重要度が増すグラフ的に与えられたプログラムの仕様から並列プログラムを自動生成し実行することに関する高速化技術の開発につながり、応用範囲の広いコンピュータサイエンス分野のコンパイラ、アルゴリズム、並列処理などの基礎技術の発展にも貢献する。

研究成果の概要（英文）：Given the specification of simulation as a large irregular graph, we have established a technique to efficiently and automatically generate a parallel high-performance simulation program.
Specifically, we firstly improved the parallelization rate of automatic program generation by dividing the graphs at a higher level of abstraction, which resulted in a 10 times speedup compared to fine-grained graph division. We secondly optimized the generated program according to a variety of parallel computers - the GPU, CPU clusters, and vector supercomputers -, which achieved an about two times speedup at most. Using the biological simulator that is one of the main applications of this work, users can get the benefits of various parallel computers without creating programs.

研究分野：高性能計算

キーワード：超高速情報処理　アルゴリズム　生体機能シミュレータ　自動並列化　データアクセス　GPU　パラメータスイープ

様　式　Ｃ－１９、Ｆ－１９－１、Ｚ－１９（共通）

１．研究開始当初の背景

過去の基盤研究（Ｂ）(相互依存関係を持つ異種タスクの同時処理に関するＧＰＧＰＵによる高速化の研究)で，異種タスクを多数同時に処理する GPGPU 技術を開発した．GPGPU(General Purpose computations on GPU)とは，コンピュータの画面描画部品である GPU (Graphic Processing Unit)の高い演算能力を，画面描画ではなく数値計算等の汎用(General Purpose)処理に適用するプログラミング技法である．この GPGPU 技術により，汎用の生体機能シミュレーションが GPU の並列処理機能により高速実行できるようになった．この技術を応用して，シミュレーションの仕様（グラフ）をもとに GPU プログラムを自動生成する生体シミュレータ Flint-on-GPU [O14]を開発した．

Flint-on-GPU により，例えば心室筋細胞の膜電位ダイナミクスを再現する生体機能モデル（以下では，FSK 心臓モデルと記す）の研究が加速し，薬物動態から心臓電気生理までの統合的にシミュレーションでき，新しい強心薬ベスナリノンの濃度に依存して心臓が正常な動きに戻ることなど多くの成果が出始めている．しかし，この研究の実施過程で，次で説明するように大規模なグラフを対象とした場合，シミュレーションプログラムの生成が長時間を要することが問題となり始めた．

一般に，科学分野のシミュレーションは次の 4 過程を繰り返す．

1) 対象を数理モデル化
2) シミュレーションプログラムの作成
3) シミュレーションの実行
4) シミュレーション結果の考察

過程 1)および 4)は対象モデル分野の研究者が行う．しかし，これらの研究者は，プログラム作りの専門家ではない．脳のようにニューロンが不規則・複雑に接続されたものの生体機能をシミュレーションする場合に，脳分野の研究者が過程 2)および 3)を行うことは容易ではない．Flint-on-GPU は，ユーザがプログラム作成にまったく関与することなく，シミュレーションの仕様をグラフで与えるだけで，過程 2)および 3)を全自動で行うことによりこの問題点を解決している．しかも，過程 3)に関しては，並列処理により高速化し，よい性能を得ている．

一方，過程 2)は自動的に行えることはよいが，モデルが大規模になると過程 2)に時間がかかり，過程 2)～3)の合計時間を考えると無視できなくなってきた．例えば，前述の FSK の場合は過程 2)だけに要する時間が 328 分（約 5 時間半）である．モデル作成の試行錯誤のため，過程 2)は多数回実施されること，およびますます大規模なモデルが対象となるので，過程 2)を高速化することがシミュレーションによる科学研究過程において重要である．

２．研究の目的

本研究の目的は，シミュレーションの仕様が大規模かつ不規則なグラフで与えられたときに、そのシミュレーションをする高速な並列プログラムを効率よく自動生成することである．

本研究の成果により，大規模で不規則かつ複雑な対象がシミュレーション可能となり，モデル作成も含めたシミュレーション過程全体の時間を短縮できる．したがって，これらのシミュレーションを必要とする多くの科学分野の研究を加速する．また，科学シミュレーションに関する貢献だけでなく，今後重要度が増すグラフ的に与えられたプログラムの仕様から高性能な並列プログラムを生成し実行することに関する高速化技術の開発につながり，応用範囲の広いコンピュータサイエンス分野のコンパイラ，アルゴリズム，並列処理などの基礎技術の発展にも貢献する．

より具体的な目的として，次の 2 つの課題に取り組む．

① シミュレーションプログラムの自動生成（過程 2)）における生成時間の短縮
② シミュレーション実行（過程 3)）における性能最適化

- GPGPU に関する既開発技術の洗練
- タイプの異なる並列計算機に関する新規の自動最適化技術

３．研究の方法

３．１ 諸定義

具体的な方法を説明するために，基本的な用語等を説明する．

1) シミュレーション内の計算順序

本研究で扱うシミュレーションにおいて，１時刻当たりのシミュレーション内容は，非常に多くの微分方程式・関数式（以降では単に式と書く）で構成され，それらの式の内容および式の計算順序の依存関係はマークアップ言語 PHML (http://physiodesigner.org/phml/index.html)で記述する．この記述全体を P とする．これらの式間の計算順序の依存関係を決定するために，P を次の性質を持つ有向グラフ $G=(V, E)$に変換する（**図１**）．ここで V は頂点の集合，E は有向辺の集合である．各頂点は式を表し，有向辺$<u, v>$は頂点 u が表す式 e(u)の計算結果を頂点 v が表す式 e(v)を計算するために必要とすることを意味する．なお，E は互いに素な次の２種類の部分集合 E_1および E_2に分割できる（$E = E_1 \cup E_2$）．時刻 t における式の結果が，同時刻 t における他の式の計算に必要であることを意味する有向辺の集合が E_1であり，時刻 t における計算結果を次の時刻 t+1 における他の式の計算に必要であることを表す有向辺の集合が E_2である．一つの時刻における計算依存関係を表すのがグラフ $G_1=<V, E_1>$となる．

2) 並列計算可能な式の抽出

$G=(V, E_1 \cup E_2)$からE_2を削除したグラフ$G_1=<V, E_1>$において，E_1の有向辺の系列（道）を解析することにより，Vを次の性質を満たす互いに素な部分集合の列 $<V_1, V_2, \ldots, V_m>$ に分割できる．I) V_1に属する頂点に入る辺はない．II)各$V_j(2 \leqq j \leqq m)$に属する頂点に入る辺は，$V_i(i < j)$に属する頂点のみから入る．II’)各V_j $(1 \leqq j \leqq m)$において，V_jの2頂点を結ぶ辺は存在しない．これらより，V_1の頂点に対応する式をすべて計算すれば，次にV_2の頂点に対応する式すべて$e(V_2)$を計算できる状況になる．同様に，$e(V_2)$を計算すれば，$e(V_3)$を計算できる状況になる．このように$e(V_1)$, $e(V_2), \ldots, e(V_m)$の順に計算することができる．さらに，II’）より各$e(V_j)(1 \leqq j \leqq m)$において，$e(V_j)$のすべての式は，独立にすなわち並列に計算できる．これが並列計算可能な式の集合$e(V_j)$を抽出するアイデアである．

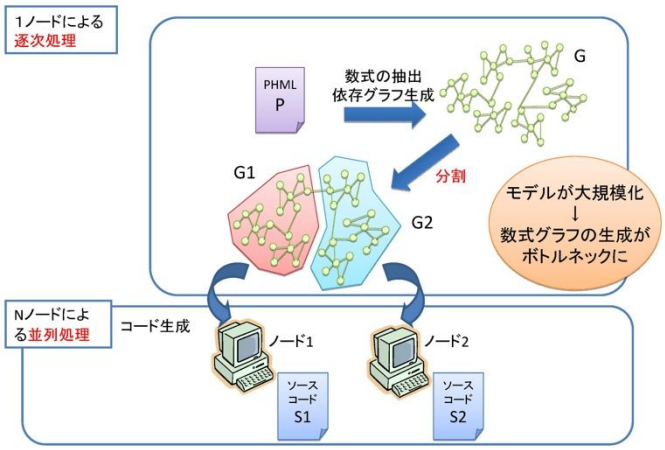

図１．細粒度グラフ分割とプログラム生成

3) 式の集合e(V)を複数のプロセッサで並列処理する手段

簡単のため2つのプロセッサP_aおよびP_bで処理するものとする．グラフ$G_1=<V, E_1>$において，Vを2つの互いに素な部分集合V_aとV_b $(=V - V_a)$に分割する．このとき，V_aの頂点とV_bの頂点を結ぶ辺をE_{1ab}とする．E_{1ab}の辺は，V_aおよびV_bに対応する式をプロセッサP_aおよびP_bで処理する場合に，他方のプロセッサでの計算結果を必要とする式の組合せである．すなわち，E_{1ab}の辺数は，ある時刻のシミュレーション中にプロセッサ間のデータ転送が必要なデータ量に相当する．並列処理をする場合，プロセッサ間のデータ転送量は少ない方が効率がよいので，E_{1ab}の辺数がなるべく少なくなるようにV_aとV_bを選べばよい．ただし，プロセッサP_aでの処理量はV_aの頂点数に比例するので，V_aとV_bは頂点数はなるべく均等となるようにVを分割するのがよい．これらのことを一般化して，複数のプロセッサそれぞれの計算量がなるべく均等で，かつプロセッサ間のデータ転送量が少ないようにVを分割するのがよい．

4) 時刻 t と t+1 の処理間のデータ依存

$G=(V, E_1 \cup E_2)$からE_1を削除したグラフ$G_2=<V, E_2>$を考える．E_1は任意の一つの時刻におけるシミュレーションで必要な式間の依存関係を表すものであるが，E_2は任意の時刻 t の式の計算と次の時刻 t+1 の式の計算の依存関係を表す．V_aの頂点とV_bの頂点を結ぶE_2の辺をE_{2ab}とする．E_{2ab}は，プロセッサP_aとP_bとの間で，次の時刻 t+1 における式の計算をするために，他方のプロセッサでの計算結果を必要とする式の組合せである．E_{1ab}と同様にE_{2ab}もなるべく辺数が少なくするようにVを分割することが好ましい．ただし，一般にE_{1ab}とE_{2ab}の両方の辺数を少なくすることは容易ではなく，グラフ理論的に興味深い問題である．

３．２ 方法

1) シミュレーションプログラムの自動生成における生成時間の短縮

生成時間が長くなる理由は，$G=(V, E_1 \cup E_2)$が非常に大規模になり，有向辺の依存処理やGの分割処理に多くの時間を必要とするからである．研究目的の欄で記載した約12万個の心筋細胞の心臓モデルFSKの場合は，約460万個の微分方程式を含み，プログラム生成時間は約５時間半となる．また，主記憶領域を約50GB使用している．モデルが大規模化すると，ますますプログラム生成時間が長くなり，主記憶領域使用量も大きくなり，この２点がボトルネックになる．

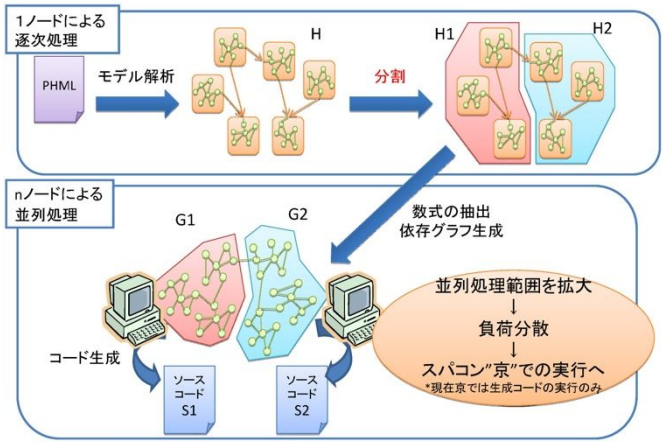

図２．抽象度の高いグラフ分割とプログラム生成法

そこでPHML記述Pから，細粒度なグラフ$G=(V, E_1 \cup E_2)$を生成するのではなく，より抽象レベルの高い疎粒度なグラフHを作成する．たとえば，上記心臓モデルFSKの場合，約460万個の微分方程式のレベルで頂点を対応させるのではなく，約12万個の心筋細胞のレベルで頂点に対応させる．すなわち，Gにおける頂点集合が，Hにおける一つの頂点に対応する．有向辺に関しては，E_1に有向辺<u, v>が属する場合，uを含む頂点集合に対応するHの頂点から，vを含む頂

点集合に対応する H の頂点に有向辺を引く．E_2 に関しても同様である（**図２**）．
利用可能なプロセッサ数を n とする．グラフ G に対して 1～2 桁規模の小さいグラフ H を n 個に分割し，それらを{H_1，H_2，…，H_n}とする．各 H_j（$1 \leqq j \leqq n$）に関する以降の処理は，n 個のプロセッサで並列に処理する．すなわち，G より小規模な H の段階でグラフ分割するので，G を分割するより高速に分割できる．さらに H_j を展開して式の依存や類似式の分類などの詳細処理をするための細粒度な G_j を作成する．図 2 に示すように，プログラム生成に占める逐次処理の割合を従来手法（図 1）と比較して小さくできるので，並列化率が向上し，ノード数に応じた速度向上を期待できる．

2) シミュレーション実行における性能最適化

従来手法が生成プログラムに施す並列化は，実行環境に依存しない汎用的な並列化手法である．現在では，並列プログラムの実行環境として多様な並列計算機が利用可能であり，シミュレーション実行をさらに高速化するためには並列計算機の特性に応じた性能最適化が不可欠である．本研究では，広く利用されている①クラスタシステム，②GPU，③ベクトルプロセッサの 3 種を対象に，それぞれのアーキテクチャを考慮して生成プログラムを自動的に洗練し，その性能を向上する．

①クラスタシステム向けの洗練手法

クラスタシステムにおいてはグラフ G の分割結果に応じて複数の計算機に処理を分散するため，分割手を洗練することで性能向上を期待できる．V の均等化と切断辺の最小化を目的とする多制約分割問題は NP 困難であり，大規模な G に対して厳密解を求めることは難しい．従来手法では，既知のヒューリスティックス *k*-way partitioning[K14]を利用して G を分割する．しかし，特定の条件を満たす G おいて V の分割に偏りが生じ，プロセッサ間で計算負荷が不均衡になる．
そこで，負荷均衡を最優先制約とし，その下で切断辺を最小化する分割手法を開発した．具体的には，*k*-way partitioning を用いてプロセッサ数 n よりも多い数 m 個にグラフ H を分割した $F=\{F_1,F_2,..,F_m\}$を得て，F_i をリストスケジューリングを用いて結合することで，{H_i ｜ $1 \leqq i \leqq n$, $H_i \subset F$）を得る．この手法の要点は，切断辺の僅かな増大と引き替えに負荷均衡を改善できる点にある．さらに，過剰な分割を避けるために，分割結果が均衡するような最小の分割数 m を自動決定する．

②GPU 向けの洗練手法

GPU においては，その高いメモリ帯域幅を活かすために，メモリ上のデータ配置と参照パターンを洗練してメモリ参照効率を向上することが肝要である．
本研究では， GPU の参照パターンを解析し，L1 キャッシュを効率的に参照するデータ配置の決定問題がハイパーグラフの分割問題に帰着できることを示した．生体モデルを示すグラフ G＝（V，E）と，G から生成した GPU プログラムの参照パターンから，ハイパーグラフ G’=(V， B)が得られる．頂点 v∈V はデータを表す．辺 b∈B は GPU 上で論理的に同時に実行されるスレッドブロック（TB）を表し，そのスレッドブロックが参照するデータ集合で定義する（つまり b ⊂V）．V を分割して得られる各クラスタ{$V_1,V_2,..,V_k$}を，1 つのキャッシュライン（グローバルメモリ上のチャンク）にまとめて置くべきデータ集合と解釈する．すると，辺 b がクラスタ Vi に架かる（図 3）ことは TB b がチャンク Vi を参照する必要があることを示す．したがって，各辺のスパン(辺と共通要素をもつクラスタ数)を最小化するようにクラスタを分割することで，TB のキャッシュへのアクセス回数を最小化できる．

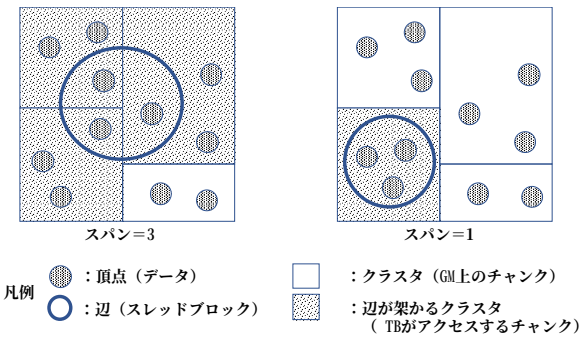

図 3. スパンと TB アクセスの関係

上記の条件を満たすハイパーグラフ分割問題は NP 困難であり，大規模な G に対して厳密解を得ることは難しい．そこで，貪欲法に類似したヒューリスティックアルゴリズムを提案し，キャッシュミスの比較的少ないデータ配置を効率的に求める手法を開発した．具体的には，以下の手順でクラスタ（隣接して配置すべきデータのサブセット）を決定する

1. 初期分割の決定：ワープと呼ばれる GPU 上の処理単位ごとに V の頂点をまとめた小クラスタ集合を作成
2. スパン減少量の予測：頂点のペアそれぞれに対して，同じクラスタに入れる際のスパンの和の減少量を推定し列挙する
3. 小クラスタの併合：スパンの和の減少量の推定値が大きい順に 2.で列挙した頂点対の属する小クラスタを併合し，解となるクラスタ集合を得る

③ベクトルプロセッサ向けの洗練手法

ベクトルプロセッサにおいて高性能を得るためには，プログラム全体に占めるベクトル演算の割合（ベクトル化効率）を最大化し，同時にメモリからベクトルを読み出す際のメモリ参照効率を最大化する必要がある．
本研究では，ベクトル型スーパーコンピュータ SX-ACE を対象に生成プログラムの洗練手法を

開発した．Flint は不規則な参照パターンを含むプログラムに対するベクトル化効率を最大化するため，配列の間接参照を多用した SX-ACE コードを生成する．データの依存関係を明示することで間接参照のベクトル化が可能であるが，直接参照するベクトル化と比較するとメモリ参照のコストが大きい．そこで，参照パターンを解析し，不必要な間接参照を自動的に削減する手法を提案した．さらに，間接参照が減少するようにデータ配置とベクトル化されるループ内の参照順を最適化した．

SX-ACE のもう 1 つの特徴として，ベクトル化の対象となる配列の定義および参照において，配列要素へ線形アクセスを行う場合にアクセス性能が高い．そこでベクトル化されるループの順番を入れ替えることでアクセスパターンを線形に変換する最適化手法を用いる．ただし，複数のベクトルデータ間で参照する配列要素に重複がある場合，線形アクセスの最大化には組み合わせ最適化を解く必要がある．そこで，メモリアクセス性能の高いアクセスパターンを貪欲法に基づいて求める手法を提案した．ベクトルデータごとの空間局所性を定量的に予測し，局所性の向上を期待できる順にループの順番と配列要素の配置を決定する．

４．研究成果

本研究の応用成果である生体シミュレータ Flint を用いて，2 つの大規模な生体モデルを対象とした実験を行った．心筋細胞の膜電位モデル（FSK）は，心房を模した 3 次元トーラス状に心室筋細胞を接続したモデルで，グラフの頂点数は 455,890，辺数は 39,585,395 である．1 次視覚野モデル（V1）は，視神経細胞を 2 層 2 次元メッシュ状に接続したモデルで，グラフの頂点数は 39,522，辺数は 31,949,922 と FSK よりも辺密度が高い．

４．１ シミュレーションプログラムの生成時間の短縮

FSK モデルを対象にした実験の結果，プログラム生成における並列化率を 0.30 から 0.89 へ向上した．すなわち，コード生成時間は並列化により逐次実行時の約 11%まで短縮可能となった．また，1 プロセッサが消費する最大主記憶量を最大 79%削減した．

４．２ クラスタシステムにおけるシミュレーションの最適化

提案手法を用いると，プロセッサの計算負荷の最大値と平均値の差を，従来手法で不均衡である場合（V1 モデル）に対して 1.30 倍から 1.01 倍に，均衡する場合（FSK モデル）に対して 1.03 倍から 1.00 倍に均衡化した．その結果，計算速度に関しては，不均衡な場合に対してマルチコア CPU で 1.18 倍，CPU クラスタで 1.22 倍の高速化を達成した．この成果は，以下の GPU および SX-ACE に対する成果と併用が可能で，GPU クラスタおよび複数の SX-ACE を結合したスーパーコンピュータを対象にした場合も性能向上を期待できる．

４．３ GPU におけるシミュレーションの最適化

NVIDIA Tesla V100（640 個の Tensor コア）を用いた FSK モデルのシミュレーションにおいて，線形アクセスの増大に特化したデータ配置と比較して，L1 キャッシュミス数を 70%に削減し，最大 1.28 倍の速度向上を達成した．ハイパーグラフ分割に要する時間は，頂点数が 2000 万を越える大規模なハイパーグラフに対して 38 分であり，シミュレーション全体の実行時間と比べて無視できるものと考えている．

４．４ SX-ACE におけるシミュレーションの最適化

不要な間接参照を削除する手法を適用した結果，　FSK モデルから生成した SX-ACE プログラムに対して，提案手法は SX-ACE 上のメモリバンド幅ピーク性能比を 15%から 20%に向上し，実行時間を最大 25%削減した．さらに，線形アクセス増大手法を併用した結果，同じ FSK モデルに対して，上記の手法と比較して線形アクセスの割合を 30%から 60%に増大し，最大 1.15 倍の速度向上を達成した．

４．５ まとめ

本研究の成果は，現時点では生体機能モデルのみに利用されているが，本質的には陽に記述した微分方程式で表現可能な対象であれば何でもよい．感染モデル，運動方程式で記述する歩行運動や腕の曲げ伸ばしなどもシミュレーションの対象となる．剛体運動，細胞膜電位モデルに使うような電気回路，タンパク相互作用のような化学反応などを統合して扱う場合は，微分方程式で陽に記述することになるので，本研究の成果の適用範囲は広い．

<引用文献>

[O14] T. Okuyama, M. Okita, T. Abe, Y. Asai, H. Kitano, T. Nomura, and K. Hagihara: Accelerating ODE-based Simulation of General and Heterogeneous Biophysical Models using a GPU, IEEE Transactions on Parallel and Distributed Systems, 25-8, 1966-1975, (2014)
[K14] G. Karypis and V. Kumar: Multilevel Algorithms for Multi-constraint Graph Partitioning, Proc. Conf. Supercomputing (SC 98), San Jose, CA, pp. 1-13 (1998).

５．主な発表論文等

〔雑誌論文〕　計11件（うち査読付論文　11件／うち国際共著　0件／うちオープンアクセス　0件）

１．著者名 Jingcheng Shen, Kentaro Shigeoka, Fumihiko Ino, and Kenichi Hagihara	４．巻 31
２．論文標題 GPU-based Branch-and-Bound Method to Solve Large 0-1 Knapsack Problems with Data-centric Strategies	５．発行年 2019年
３．雑誌名 Concurrency and Computation: Practice and Experience	６．最初と最後の頁 1-15
掲載論文のDOI（デジタルオブジェクト識別子） 10.1002/cpe.4954	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Nobuhiro Miki, Fumihiko Ino, and Kenichi Hagihara	４．巻 13
２．論文標題 PACC: A Directive-based Programming Framework for Out-of-Core Stencil Computation on Accelerators	５．発行年 2019年
３．雑誌名 International Journal of High Performance Computing and Networking	６．最初と最後の頁 19-34
掲載論文のDOI（デジタルオブジェクト識別子） 10.1504/IJHPCN.2019.097046	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Yasuaki Mitani, Fumihiko Ino, and Kenichi Hagihara	４．巻 29
２．論文標題 Parallelizing Exact and Approximate String Matching via Inclusive Scan on a GPU'	５．発行年 2017年
３．雑誌名 IEEE Transactions on Parallel and Distributed Systems	６．最初と最後の頁 1989-2002
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/TPDS.2016.2645222	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

１．著者名 Yuechao Lu, Fumihiko Ino, and Kenichi Hagihara	４．巻 E99-D
２．論文標題 Cache-Aware GPU Optimization for Out-of-Core Cone Beam CT Reconstruction of High-Resolution Volumes	５．発行年 2016年
３．雑誌名 IEICE Transactions on Information and Systems	６．最初と最後の頁 452-461
掲載論文のDOI（デジタルオブジェクト識別子） 10.1587/transif.2016EDP7174	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Yuji Misaki, Fumihiko Ino, and Kenichi Hagihara	4．巻 E100-D
2．論文標題 Cache-aware, In-place Rotation Method for Texture-based Volume Rendering	5．発行年 2017年
3．雑誌名 IEICE Transactions on Information and Systems	6．最初と最後の頁 3060-3071
掲載論文のDOI（デジタルオブジェクト識別子） 10.1587/transinf.2016EDP7178	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Yasuaki Mitani, Fumihiko Ino, and Kenichi Hagihara.	4．巻 28
2．論文標題 Parallelizing Exact and Approximate String Matching via Inclusive Scan on a GPU	5．発行年 2017年
3．雑誌名 IEEE Transactions on Parallel and Distributed Systems	6．最初と最後の頁 1989-2002
掲載論文のDOI（デジタルオブジェクト識別子） 10.1109/TPDS.2016.2645222	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Takuya Ikuzawa, Fumihiko Ino, and Kenichi Hagihara	4．巻 93/94
2．論文標題 Reducing Memory Usage by the Lifting-based Discrete Wavelet Transform with a Unified Buffer on a GPU	5．発行年 2016年
3．雑誌名 Journal of Parallel and Distributed Computing	6．最初と最後の頁 44-55
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 三谷康晃，伊野文彦，萩原兼一	4．巻 J98-D
2．論文標題 GPU向けの反復型グラフ処理フレームワークにおけるトポロジ変更の実現	5．発行年 2015年
3．雑誌名 電子情報通信学会論文誌	6．最初と最後の頁 1365--1373
掲載論文のDOI（デジタルオブジェクト識別子） なし	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Daiki Okada, Fumihiko Ino, and Kenichi Hagihara	4．巻 16
2．論文標題 Accelerating the Smith-Waterman Algorithm with an Interpair Pruning Method for All-Pairs Comparison of Base Sequences	5．発行年 2015年
3．雑誌名 BMC Bioinformatics	6．最初と最後の頁 1-15
掲載論文のDOI（デジタルオブジェクト識別子） http://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-015-0744-4	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Shohei Ando, Fumihiko Ino, Toru Fujiwara, and Kenichi Hagihara	4．巻 5
2．論文標題 Enumerating Joint Weight of a Binary Linear Code Using Parallel Architectures: multi-core CPUs and GPUs	5．発行年 2015年
3．雑誌名 International Journal of Networking and Computing	6．最初と最後の頁 290-303
掲載論文のDOI（デジタルオブジェクト識別子） https://www.jstage.jst.go.jp/article/ijnc/5/2/5_290/_article	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

1．著者名 Ko Kusudo, Fumihiko Ino, and Kenichi Hagihara	4．巻 76
2．論文標題 A Bit-Parallel Algorithm for Searching Multiple Patterns with Various Lengths	5．発行年 2015年
3．雑誌名 Journal of Parallel and Distributed Computing	6．最初と最後の頁 49-57
掲載論文のDOI（デジタルオブジェクト識別子） http://www.sciencedirect.com/science/article/pii/S074373151400210X	査読の有無 有
オープンアクセス オープンアクセスではない、又はオープンアクセスが困難	国際共著 －

〔学会発表〕　計27件（うち招待講演　0件／うち国際学会　10件）

1．発表者名 Keishi Tsukada and Fumihiko Ino
2．発表標題 A Method for Estimating Task Granularity for Automating GPU Cycle Sharing
3．学会等名 the 7th International Conference on Network, Communication and Computing（国際学会）
4．発表年 2018年

１．発表者名
比嘉慎哉，置田真生，萩原兼一，伊野文彦

２．発表標題
GPUプログラムにおける静的参照関係を表すハイパーグラフの分割を用いた参照効率のよいデータ配置

３．学会等名
情報処理学会ハイパフォーマンスコンピューティング研究会

４．発表年
2019年

１．発表者名
石田祐二郎，置田真生，伊野文彦，萩原兼一

２．発表標題
並列プログラム自動生成における間接参照の削減によるベクトル計算機向けメモリ参照効率化

３．学会等名
第14回情報科学ワークショップ

４．発表年
2018年

１．発表者名
沈靖程，重岡謙太郎，伊野文彦，萩原兼一

２．発表標題
GPU上で大規模ナップザック問題を解くためのアウトオブコア計算手法

３．学会等名
第14回情報科学ワークショップ

４．発表年
2018年

１．発表者名
石田祐二郎，置田真生，伊野文彦，萩原兼一

２．発表標題
並列プログラム自動生成におけるベクトル計算機向けメモリ参照効率化のための間接参照の削減

３．学会等名
電子情報通信学会コンピュータシステム研究会

４．発表年
2018年

１．発表者名
Jingcheng Shen, Kentaro Shigeoka Fumihiko Ino, and Kenichi Hagihara

２．発表標題
An Out-of-core CPU-GPU Cooperative B&B Solver for the Large KnapsackProblem

３．学会等名
The 2nd Cross-disciplinary Workshop on Computing Systems,Infrastructures, and Programming (xSIG 2018),

４．発表年
2018年

１．発表者名
Yuechao Lu, Fumihiko Ino, Yasuyuki Matsushita, and Kenichi Hagihara

２．発表標題
RLAGPU: High-performance Out-of-Core Randomized Singular Value Decomposition on GPU

３．学会等名
the 8th GPU Technology Conference (GTC 2017)（国際学会）

４．発表年
2017年

１．発表者名
Fumihiko Ino, Yasuaki Mitani, and Kenichi Hagihara

２．発表標題
cuShiftOr: String Matching with Prefix Summing on a GPU

３．学会等名
the 8th GPU Technology Conference (GTC 2017)（国際学会）

４．発表年
2017年

１．発表者名
Jingcheng Shen, Kentaro Shigeoka Fumihiko Ino, and Kenichi Hagihara

２．発表標題
An Out-of-Core Branch and Bound for Solving the 0-1 Knapsack Problem on a GPU

３．学会等名
the 17th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2017)（国際学会）

４．発表年
2017年

１．発表者名
Kazuki Yasui and Fumihiko Ino

２．発表標題
Accelerating Scoring Computation of Smith-Waterman Algorithm with Mixed Word Length

３．学会等名
the 4th International Workshop on High Performance Computing on Bioinformatics (HPCB 2017)（国際学会）

４．発表年
2017年

１．発表者名
Masato Ito and Fumihiko Ino

２．発表標題
An Automated Method for Generating Training Sets for Deep Learning Based Image Registration

３．学会等名
the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018)（国際学会）

４．発表年
2018年

１．発表者名
Ryo Asai, Masao Okita, Fumihiko Ino, and Kenichi Hagihara

２．発表標題
Transparent Avoidance of Redundant Data Transfer on GPU-enabled Apache Spark

３．学会等名
the 11th Workshop on General Purpose Processing Using GPU (GPGPU 2018)（国際学会）

４．発表年
2018年

１．発表者名
Ryotaro Sakai, Fumihiko Ino, and Kenichi Hagihara.

２．発表標題
Towards Automating Multi-dimensional Data Decomposition for Executing a Single-GPU Code on a Multi-GPU System

３．学会等名
4th International Symposium on Networking and Computing Systems and Architectures(CSA 2016)(国際学会)

４．発表年
2016年

１．発表者名
Nobuhiro Miki, Fumihiko Ino, and Kenichi Hagihara.

２．発表標題
An Extension of OpenACC Directives for Out-of-Core Stencil Computation with Temporal Blocking

３．学会等名
3rd Workshop on Accelerator Programming Using Directives (WACCPD 2016), pp. 36--45, Salt Lake City, UT, USA, (2016-11).

４．発表年
2016年

１．発表者名
三木脩弘，伊野文彦，萩原兼一

２．発表標題
アウトオブコア・ステンシル計算に対する自動テンポラルブロッキングのためのアクセラレータ向けディレクティブPACC

３．学会等名
GTC Japan 2016

４．発表年
2016年

１．発表者名
酒井亮太郎，伊野文彦，萩原兼一．

２．発表標題
単一GPUコードをマルチGPU環境で実行するための多次元データ分割手法の検討

３．学会等名
情報処理学会ハイパフォーマンスコンピューティング研究会

４．発表年
2016年

１．発表者名
三木脩弘，伊野文彦，萩原兼一．

２．発表標題
アウトオブコア・ステンシル計算に対する自動テンポラルブロッキングのためのアクセラレータ向けディレクティブ

３．学会等名
情報処理学会ハイパフォーマンスコンピューティング研究会

４．発表年
2016年

１．発表者名
塚田敬司，伊野文彦，萩原兼一.

２．発表標題
GPUサイクル共有を自動化するためのタスク粒度推定手法の検討

３．学会等名
Cross-disciplinary Workshop on Computing Systems, Infrastructures, and Programming (xSIG 2017)

４．発表年
2017年

１．発表者名
安井一貴，伊野文彦，萩原兼一..

２．発表標題
GPU上の16ビット整数表現によるSmith-Watermanアルゴリズムの高速化の検討

３．学会等名
第16回ハイパフォーマンスコンピューティングと計算科学シンポジウム

４．発表年
2016年

１．発表者名
Nobuhiro Miki, Fumihiko Ino, and Kenichi Hagihara

２．発表標題
Applying Temporal Blocking to Out-of-Core Stencil Computation with OpenACC

３．学会等名
Proceedings of the Work in Progress Session held in connection with the 24th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2016)（国際学会）

４．発表年
2016年

１．発表者名
Kei Ikeda, Fumihiko Ino, and Kenichi Hagihara

２．発表標題
An OpenACC Optimizer for Accelerating Histogram Computation on a GPU

３．学会等名
Proceedings of the 24th Euromicro International Conference on Parallel, Distributed and Network-Based Computing (PDP 2016)（国際学会）

４．発表年
2016年

１．発表者名
Ryotaro Sakai, Fumihiko Ino, and Kenichi Hagihara.

２．発表標題
Preliminary Estimation on Automating Multi-dimensional Data Decomposition for Multi-GPU Systems

３．学会等名
Poster in the 2nd Annual Meeting on Advanced Computing System and Infrastructure (ACSI 2016)（国際学会）

４．発表年
2016年

１．発表者名
三木脩弘，伊野文彦，萩原兼一

２．発表標題
OpenACCを用いたアウトオブコア・ステンシル計算に対するテンポラルブロッキングの適用

３．学会等名
Poster in the 5th GPU Technology Conference (GTC Japan 2015)

４．発表年
2015年

１．発表者名
三谷康晃，伊野文彦，萩原兼一

２．発表標題
文字列探索アルゴリズムShift-Or法の結合性を利用した高速化

３．学会等名
Poster in the 5th GPU Technology Conference (GTC Japan 2015)

４．発表年
2015年

１．発表者名
三木脩弘，伊野文彦，萩原兼一.

２．発表標題
OpenACCを用いたアウトオブコア・ステンシル計算に対するテンポラルブロッキングの適用

３．学会等名
情報処理学会研究報告HPC研究会

４．発表年
2015年

１．発表者名
三谷康晃，伊野文彦，萩原兼一

２．発表標題
接頭部和の計算に基づく近似文字列探索の並列化

３．学会等名
電子情報通信学会技術研究報告CPSY研究会

４．発表年
2015年

１．発表者名
池田圭，伊野文彦，萩原兼一

２．発表標題
ヒストグラム生成を高速化するためのOpenACCオプティマイザの検討

３．学会等名
情報処理学会研究報告HPC研究会

４．発表年
2015年

〔図書〕　計0件

〔産業財産権〕

〔その他〕

大阪大学 大学院情報科学研究科 コンピュータサイエンス専攻 並列処理工学講座
http://www-hagi.ist.osaka-u.ac.jp/
汎用生体シミュレータの高速化に関する研究
http://www-hagi.ist.osaka-u.ac.jp/research/SIM/index.html

６．研究組織

	氏名 （ローマ字氏名） （研究者番号）	所属研究機関・部局・職 （機関番号）	備考
研究分担者	伊野　文彦 (Ino Fumihiko) (90346172)	大阪大学・情報科学研究科・教授 (14401)	

６．研究組織（つづき）

	氏名 （研究者番号）	所属研究機関・部局・職 （機関番号）	備考
研究分担者	置田　真生 (Okita Masao) (50563988)	大阪大学・情報科学研究科・准教授 (14401)	