修士論文

題目

GPUにおける
大規模避難シミュレーションの
エージェント特性を用いた高速化

指導教員

大野 和彦講師

2020年

三重大学大学院 工学研究科 情報工学専攻
コンピュータアーキテクチャ研究室

柴田 昂輝(418M515)

GPUにおける大規模シミュレーションのエージェント特性を用いた高速化

柴田 昂輝

内容梗概

災害時の最適な避難行動を求めるために，マルチエージェントシミュレーションが多く用いられている．マルチエージェントシミュレーションは，エージェントと呼ばれる個体を主体とすることで，ボトムアップ的に現実に近い形のシミュレーションを行う手法である．エージェントは，周囲の環境や他のエージェントと相互作用を行うことで自身の行動を決定する．しかし，実用的な避難シミュレーションは一般的に大規模である．そのため，全てのエージェントが相互作用計算を行うマルチエージェントシミュレーションは，エージェント数の増加によってシミュレーション時間が非常に長くなってしまうという問題がある．

近年，高い演算性能を持つ Graphics Processing Unit(GPU) を利用した並列処理による高速化が注目されている．GPU はコアを大量に搭載しており，多くのスレッドを起動して並列計算を行う．さらに，全スレッドを Warp と呼ばれる 32 スレッドごとに分割し，Warp に対して一つの命令を実行する形式を採用することで，効率的な並列処理を実現している．しかし，分岐処理などによって Warp 内のスレッド間で実行される命令が異なる場合，GPU の実行性能が低下してしまうブランチダイバージェンスという問題がある．マルチエージェントシミュレーションは，エージェントごとに異なる行動規則を記述することが可能であるため，分岐構造が多く存在する．したがって，ブランチダイバージェンスによる性能低下が大きい．

従来研究では，エージェントの内部状態によってエージェントをグループ化することによって，一つの Warp 内に同じ命令を実行するスレッドを割り当てる手法が提案されている．これにより，ブランチダイバージェンスを低減し，マルチエージェントシミュレーションの高速化が実現されている．しかし，エージェントの内部状態によるグループ化のみでは，人間のように複雑な意思決定を行うエージェントの行動を網羅することができない．そのため，ブランチダイバージェンスを低減しきれておらず，十分な速度向上が得られていない．

そこで本研究では，エージェントの内部状態だけでなく，エージェントの性格に基づく類型や，エージェントに付与される属性など，エージェ

ントの行動に影響を与えるようなエージェント特性に基づき，ソートを用いてグループ化する．さらに，ある類型が特定の状態の場合に行動を起こすといったような，エージェントの行動に複数の要因が同時に影響を与える場合，エージェント特性を組み合わせてグループ化する．また，シミュレーションの進行に伴い，相互作用が行われた結果，効果的なエージェント特性は変化すると考えられる．そのため，シミュレーションのステップ数や避難しているエージェントの数といった，シミュレーション環境から得られる情報に基づいて，グループ化に用いるエージェント特性をシミュレーション中に変更することで，更なる最適化を行う．エージェント数や類型の種類数，類型間でのエージェント数の偏りを変更し，それぞれの条件でソートを用いたグループ化による実験を行った．提案手法によるシミュレーション結果と状態によるグループ化の結果を比較すると，どの条件の場合でも約 65,000 エージェントを超える場合に高速化が確認でき，最大で約 62 %の実行時間削減が確認された．

Optimization of Large-Scale Evacuation Simulation on GPU using Agents Characteristics

Kouki Shibata

Abstract

Multi-Agent Simulation (MAS) is often used to explore efficient disaster evacuation. MAS regards individual evacuees as autonomous *Agents*, and simulates their behaviors by a bottom-up approach. Thereby, MAS simulates realistic behavior of the evacuees crowd. Each agent decides their next behavior during the interaction with the surrounding environment and the nearby agents. Thus the computation cost of inter-agents interaction increases significantly in practical evacuation simulations with large number of evacuees. Therefore, the acceleration using Graphics Processing Unit (GPU) is attracting attention. A GPU is equipped with thousands of cores and performs highly-parallel computation launching many threads. All threads are divided into units called Warps by 32 threads. A Warp is the minimum unit of independent execution control: each threads in a Warp executes the same instruction simultaneously. Although GPU achieves high performance by highly-parallel processing, the performance may decrease due to conditional branches. Such inefficiency is called *branch divergence* and often non-negligible in MAS because the code determinating the agent's behavior usually includes many conditional branches. The branch divergence can be solved by assigning threads executing the same program path. A previous research proposed a method grouping agents by the agent states, which reduces the branch divergence and achieves better acceleration of MAS. However, the performance is still insufficient because the grouping is not optimal. In this research, we propose a new method of grouping the agents. In addition to the agent state, we also consider agent characteristics which affect the agent behaviors. To improve the adequency of agent grouping, effective states and characteristics are selected and used in combination. As a further optimization, the selection is dynamically changed considering the simulation progress.

三重大学大学院　工学研究科

We compared the proposed method with the method considering only the agent states. As a result, the proposed method improved the performance when the number of agents is 65,000 or more, reducing the execution time by up to 62%.

目 次

図 目 次

表 目 次

1　はじめに

近年，災害避難シミュレーションではマルチエージェントシミュレーションが多く用いられている．マルチエージェントシミュレーションとは，エージェントと呼ばれる個体を主体としてボトムアップ的に現象を創発するシミュレーションである．図 1.1 はマルチエージェントシミュレー

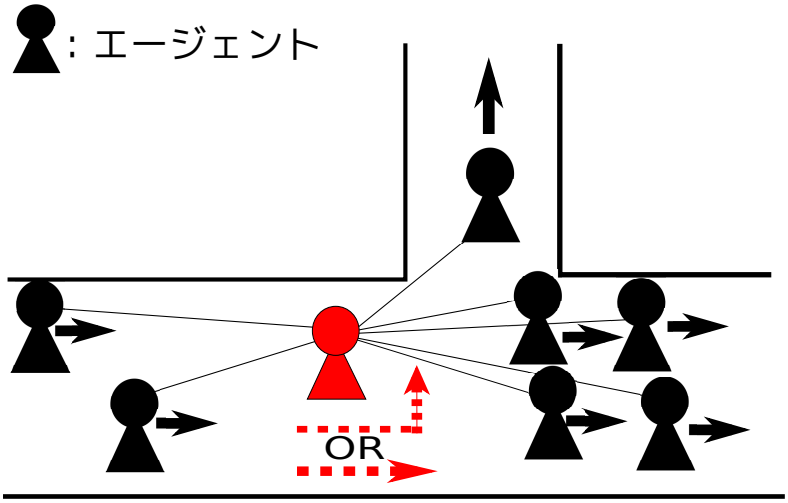

図 1.1: マルチエージェントシミュレーションの概略図

ションによる避難シミュレーションの概略を示しており，赤色のエージェントに注目している．エージェントは周囲から情報を受け取り，自身が持つ規則に従って次の行動を決定する．このように，群集行動は周囲の状況に応じて各個人の行動が変化し，それぞれがもたらす小さな変化の積み重ねによって結果が大きく変化する．以上で述べた複雑系の事象を，マルチエージェントシミュレーションは表現できる．

一般に，実用的な避難シミュレーションは大規模であり，多数のエージェントをシミュレートする．マルチエージェントシミュレーションはエージェント毎に周囲のエージェントや環境と相互作用を行うため，エージェント数が増加するとシミュレーション時間が長くなる．さらに，計算機の高性能化に伴い，エージェントがより現実的な行動を取るために，エージェント個体の多様化 [1] や思考アルゴリズムの複雑化 [2] が進んでいる．

このような計算量の増加に対し，Graphics Processing Unit（GPU）を用いた並列処理による高速化が求められている．GPU は多数のコアを搭載しており，それぞれのコアでスレッドを起動し，タスクを割り当てて並列的に処理を行うことで高い処理性能を実現している．GPU 上でマルチエージェントシミュレーションを行う場合，1 スレッドに 1 エージェントの処理を割り当てる．全スレッドは 32 スレッド毎に Warp という単位に分割され，Warp 内の 32 スレッドに対して，一つの命令が Single Instruction Multiple Data（SIMD）形式で実行される．この時，同一 Warp 内のスレッドで分岐が発生する場合，GPU の実行性能が低下するブランチダイバージェンスという問題がある [3]．この問題に対し，エージェントの状態を用いることで Warp 内の分岐を低減する手法が提案されている [15, 16]．

しかし，エージェントの行動は状態以外の様々な要素によって変化するため，エージェントの状態のみを用いただけではWarp内の分岐を低減しきれていない．

そこで本論文では，エージェントの性格・状態・属性といった，エージェントの行動決定に関わる要素を用いてエージェントデータの再配置を行い，Warp内の分岐を低減する手法を提案する．ここで，エージェントの性格・状態・属性といった，エージェントの行動決定に関わる要素を本論文ではエージェント特性と呼ぶこととする．エージェントが持つエージェント特性の中からキーとする要素を選択し，その要素を基にエージェントデータをソートすることによって，同様の行動を行うエージェントを同じWarp内に割り当てられるよう配置する．さらに，エージェントの行動に複数のエージェント特性が同時に影響を与える場合，エージェント特性を組み合わせることで新たにキーとなる要素を作成する．また，エージェントの意思決定は周囲の環境やエージェントから得る情報によって変化するため，ブランチダイバージェンスの低減に効果的なエージェント特性がシミュレーションの進行によって変化する．そこで，シミュレーション中の環境情報を利用して，ソートに用いるエージェント特性を実行時に切り替えることにより，シミュレーション進行によって新たに発

生するブランチダイバージェンスを低減する．

本論文は次のように構成されている．まず2章で，マルチエージェントシミュレーションとGPUの詳細について述べる．3章では，提案手法に関連する避難シミュレーションの研究と，GPUでの高速化に関する研究について述べる．4章では，提案手法について述べる．5章では，提案手法を評価するための実験内容と評価結果について述べる．6章で評価結果に対する考察を述べ，最後に7章でまとめと今後の課題について述べる．

2　研究背景

本章では，本研究について説明するうえで必要となる研究背景を述べる．2.1節では，マルチエージェントシミュレーションについて簡易的なモデルを例に挙げて説明する．2.2節では，マルチエージェントシミュレーションの並列化に用いられるGPUについて述べる．

2.1　マルチエージェントシミュレーション

マルチエージェントシミュレーションは，エージェントと呼ばれる個体を主体として行うシミュレーション手法である．各エージェントは周囲のエージェントや環境から情報を取得し，その情報に基づいて自身が次に行う動作を決定する．このような相互作用によって，エージェントの

動作をシミュレートすることで，ボトムアップ的に全体の現象を創発している．図 2.2 はエージェントの相互作用の例を示しており，赤色のエー

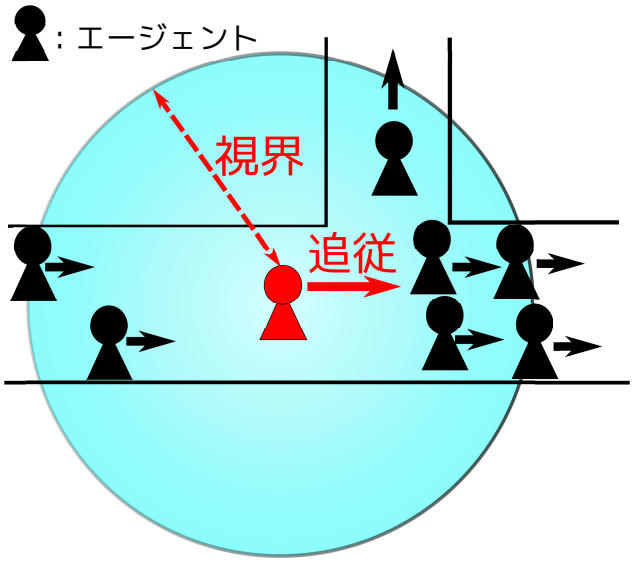

図 2.2: エージェントの相互作用

ジェントに注目している．注目エージェントは円状の視界を持っており，その範囲内の他エージェントや環境と相互作用を行なっている．注目エージェントが分かれ道で多数のエージェントに追従する性質を持っている場合，視界内の他エージェントが多い方向に進行方向を決定する．このように，周囲の状況に応じて各個人の行動が変化し，それぞれがもたらす小さな変化の積み重ねによって，大きく結果が変化する複雑系の事象を表現することが可能である．

エージェントのモデルには静的な属性や動的に変化する状態が与えら

れる．図 2.3 はエージェント内部データの具体例を示している．このエージェントは子ども属性を有しており，迷子状態に陥っている．この時，子どもエージェントは親エージェントを探して歩き回るといった行動を取る．このようにマルチエージェントシミュレーションでは，エージェン

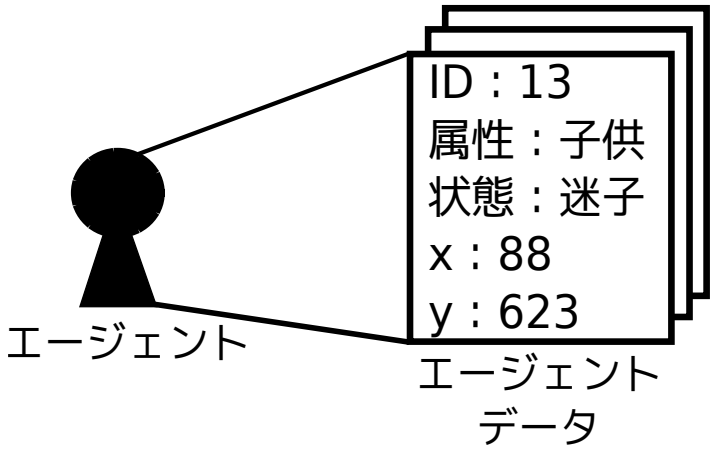

図 2.3: エージェントデータの具体例

トの属性によってエージェントの性質を定め，エージェントの状態によってエージェントの行動が条件付けられる [4]．さらに，獲物と捕食者の行動を表現したモデル [5] では，エージェントを獲物と捕食者という静的な属性に分けている．獲物と捕食者の状態は有限オートマトンによって動的に管理されており，獲物が捕食者を見つけるか，捕食者が獲物を見つけることによって状態が遷移する．この状態遷移により，獲物は安全な巣穴を探し，捕食者は獲物を追いかける，といった状態に応じた行動を取るようになる．

以上のように，マルチエージェントシミュレーションは相互作用によって創発される現象を表現できる．しかし，エージェント同士の相互作用を毎ステップ行うため，エージェント数が増加すると計算量が多くなり，シミュレーションにかかる時間が増大する問題がある．また，従来は上で述べたような単純なモデルによるシミュレーションを行なっていたが，計算機の高性能化に伴い，より現実的なエージェント動作のモデル化も行われている．このような計算量の増加に対し，GPU を用いた並列処理による高速化が求められている．

2.2 GPU

本節では，GPU の概要と並列処理を行う際の問題点を挙げる．なお本研究では，NVIDIA 社製 GPU とその開発環境である CUDA を用いている．CUDA は C/C++を拡張し，NVIDIA 社製の GPU に最適化されているため，GPU 性能を引き出すことが可能である．また，ライブラリやドキュメントも充実している [3] ことから，利用しているユーザが多い．

2.2.1 概要

GPU は数百から数千のコアを搭載しており，Streaming Multiprocessor（SM）によって効率的に実行されるよう管理されている．1 つの SM には

最大 1024 本のスレッドが割り当てられ，並列に実行される．GPU 上でマルチエージェントシミュレーションを行う場合，1 スレッドには 1 エージェントの処理を割り当てるため，非常に多くのエージェントの処理を並列化することが可能である．また，SM は全てのスレッドを 32 スレッド毎の Warp という単位に分割する．GPU は Warp を命令の最小実行単位として，Warp 内の全スレッドに一つの命令を発行する形式で処理を行なっている．図 2.4 に，GPU における Warp 内の命令実行方式を示す．図 2.4 は大きさが 32 の配列 A,B,C に対して，$A[i] + B[i] = c[i]$ $(0 \leq i < 32)$ という計算を 32 スレッドで行う場合の処理を示している．スレッドはスレッド番号という一意の通し番号を有しており，この番号を用いて各配列の要素にアクセスしている．これにより，Warp 内の各スレッドが配列 1 要素分の計算を独立して行っている．この実行方式により，32 回の加算を必要とする処理を，一度の加算で並列に実行している．

一方，条件分岐などによって同一 Warp 内の 32 スレッド間で実行される命令が異なる場合，GPU の実行性能が低下するブランチダイバージェンスという問題が発生する．ブランチダイバージェンスについては，2.2.2 項で詳しく説明する．

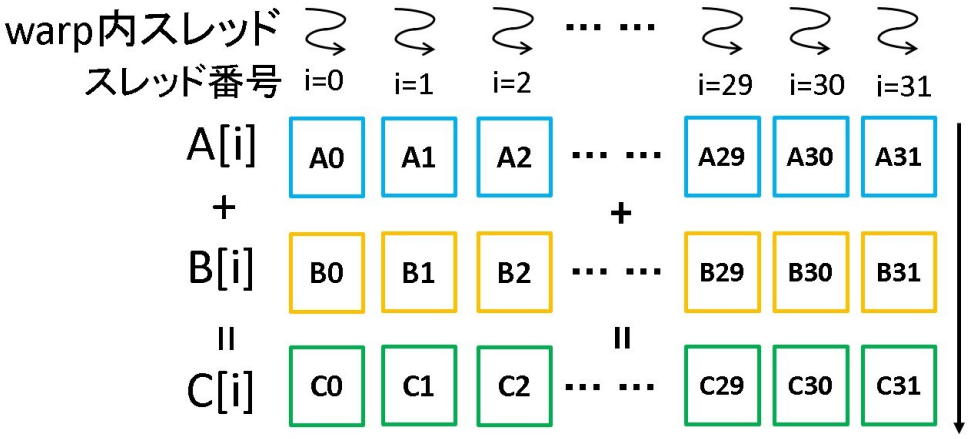

図 2.4: Warp 内の命令実行方式

2.2.2 ブランチダイバージェンス

2.2.1 項で述べたように，同じ Warp 内のスレッドは一つの命令しか実行することができない．そのため，同じ Warp 内のスレッドが条件分岐などによって異なる命令を実行する場合，それぞれの実行パスが逐次的に実行される．図 2.5 にブランチダイバージェンスの例を示す．右の四角は Warp を示しており，赤い矢印が動作を行なっているスレッドを，灰色の矢印がアイドル状態になっているスレッドを示している．また紙面の都合上ここでは，12 本のスレッドによって Warp が構成されているものとする．

図中の Step2 の条件分岐によって，Warp 内のスレッドが If 節と Else 節

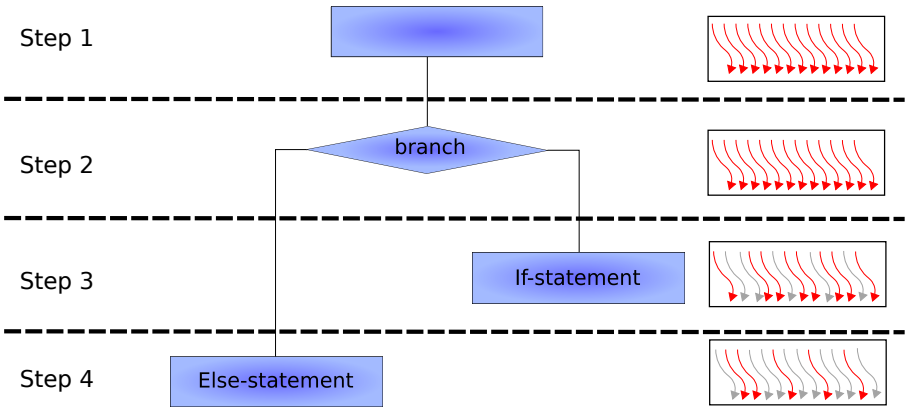

図 2.5: ブランチダイバージェンス

の 2 つの実行パスに分岐したとする．この場合，ブランチダイバージェンスが発生し，If 節と Else 節が逐次的に実行される．図中の Step3 で，If 節の処理を実行している間は，Else 節を通るスレッドが割り当てられているコアはアイドル状態になる．この時，Warp 内の 12 スレッド中 8 スレッドしか稼働していないため，Warp に割り当てられているコアの稼働率が 67%(8/12) に低下する．同様に，図中の Step4 で Else 節の処理を実行している間は，If 節を通るスレッドが割り当てられているコアはアイドル状態になる．この時，Warp 内の 12 スレッド中 4 スレッドしか稼働していないため，Warp に割り当てられているコアの稼働率が 33%(4/12) に低下する．このように，ブランチダイバージェンスが原因で，コアがアイドル状態になってしまうことによって，GPU の性能低下を引き起こしている．

マルチエージェントシミュレーションはエージェントの行動制御に多くの分岐が存在するため，ブランチダイバージェンスによる性能低下が大きい．そのため，ブランチダイバージェンスを低減する研究が行われており，その中のいくつかの関連研究については，3.2 節で述べる．

3　関連研究

本章では，マルチエージェント方式の避難シミュレーションに関する研究と，GPU 上での高速化に関する研究について述べる．

3.1　避難シミュレーション

本節では，マルチエージェント方式の避難シミュレーションに関する研究について述べる．3.1.1 項では，エージェントの局所的な行動モデルについての研究について述べる．3.1.2 項では，より現実的なエージェントの行動モデルについての研究について述べる．

3.1.1　エージェントの局所行動モデル

マルチエージェントシミュレーションでは，周囲の環境やエージェントとの間の相互作用の積み重ねによって事象を創発している．そのため，エージェントの局所的な行動は全体として大きく事象の表現に関わって

くることから，どうモデル化するかという問題は重要であり，多くの研究が続けられている．

歩行者の行動を物理学的な側面からモデル化した研究として，Social Force Model[6] がある．このモデルでは，歩行者が回避行動やルート変更を行う時の意志や感覚をベクトル量として表現している．例えば，前方から歩いてくる歩行者に対しては避けようとする斥力が働き，目的地に対してはその地点に向かおうとする引力が働く．これら複数のベクトル量を合計したものを，歩行者が受ける外力とし，歩行者の行動ルールを運動方程式に組み込むことで，エージェントが移動する方向を決定している．

一方，歩行者の特性をより現実に即した方式で実装した歩行者シミュレーションシステムとして，Sim Tread[7] がある．このシステムでは，全歩行エージェントが一定間隔毎に,「仮移動位置の計算」・「衝突回避」・「仮移動位置へ移動」の三つの計算を繰り返し行なっている．仮移動位置の計算では，全ての歩行者が目的地へ最も近づく位置を計算する．衝突回避の計算では，エージェントごとに定められたエージェントの寸法と衝突判定領域内に，障害物や他の歩行者が存在する場合に，進む方向や速さを変更しながら仮移動位置を一定回数まで再計算する．一定回数，衝

突回避計算を繰り返しても適切な位置へ移動が出来なかった場合，該当エージェントは移動しない．このような計算手順を経ることにより，歩行者が衝突を回避する過程を忠実に表現している．

3.1.2　エージェントの現実的な行動モデル

マルチエージェントシミュレーションは，各エージェントに異なる属性や行動規則を定義することができる．それにより，自身の役割や周囲の状況に応じて行動が変化する．しかし，2.1 節で挙げたような簡単な行動規則のみのマルチエージェントシミュレーションでは，現実の避難者の動きを表現することが出来ない．そこで，エージェントの行動をより現実に近いものとするための研究が進められている．

様々な役割を持つエージェントをシミュレートするものに，ロボカップレスキューのレスキューシミュレーション [8] がある．ロボカップレスキューでは，大都市で災害が発生した状況を想定し，都市が受ける被害をできるだけ小さくすることを目的としている．このモデルでは，避難者エージェントだけでなく，救助エージェント・消防エージェント・土木エージェントなども仮想環境に投入して，シミュレーションを行なっている．救助エージェントは，建物に取り残されたり，倒壊に巻き込まれたりした要救助者エージェントを捜索して救助活動に当たる．消防エー

ジェントは，各地の火災の状況を考慮して人員配分を考慮しながら消火活動に当たる．土木エージェントは，建物の倒壊などによって生じた瓦礫を撤去し，消防エージェントの活動を手助けする．このような役割の異なるエージェントは，動作が大きく異なる可能性があるため，分岐によるブランチダイバージェンスの影響が大きい．

また，避難者や誘導員などの音声による情報伝達行為を考慮したマルチエージェントモデル [9] がある．このモデルでは，目的地情報を持たないエージェントは付近のエージェントに目的地情報を尋ね，付近のエージェントが目的地情報を有している場合，情報提供を行う．このように，同じ避難者エージェントであっても，目的地情報を持っている状態か，持っていない状態かによって行動が変化するため，ブランチダイバージェンスが発生してしまう．

エージェントの内部状態によって行動が変化するものとして，エージェントに心理モデルを導入した研究 [10] がある．このモデルでは，避難時に見られる行動を引き起こす心理状態を設定し，この心理状態を引き起こす 4 つの心理要因を定義することによって，被災者の心理モデルを構築している．この心理要因に関連するパラメータはエージェントごとに値が設定されている．そして，シミュレーション中に周囲の環境から得

られた情報によって心理要因が活性化し，心理状態が決定される．これにより，エージェントは心理状態に対応した災害時の行動をとるようになる．最初に設定される各エージェントの心理要因に関するパラメータは，そのエージェントが避難時に取る行動を傾向づけるものであり，エージェントの性格と捉えることができると考える．

避難者の行動傾向から，エージェントを分類してモデル化した研究 [11] がある．この研究では，サーベイリサーチセンターが行なった東日本大震災の被災者に対するアンケート [12] を参考に避難者の行動傾向を類型化し，エージェントを類型に分けてモデル化している．それぞれの類型によって，避難に対する積極性が異なり，それに対応するように避難開始の条件や避難中に取る行動が異なる．

上で述べたような，エージェントの行動をより現実に近づけるようなモデルは，エージェントの意思決定がより複雑化している．そのため，複数の要素が影響し合って行動が変化する場合も存在している．

3.2 GPU

本節では，GPU について本研究と関連する研究について述べる．3.2.1 項では，ブランチダイバージェンスの低減に関する研究について述べる．

3.2.2 項では，GPU 上のマルチエージェントシミュレーションに対して，ブランチダイバージェンスを低減する手法を適用している研究について述べる．

3.2.1 ブランチダイバージェンス

GPU において，ブランチダイバージェンスによる性能低下は影響が大きく，ブランチダイバージェンスを解決するために様々な提案がされている．

ブランチダイバージェンスが発生した際の性能低下を抑える研究 [13] がある．この研究では，条件分岐によって生じる分岐パスから類似している部分を抜き出し，分岐パスの外に追い出す手法を提案している．このように，分岐先のサイズを小さくし，分岐先の実行サイクル数を減らすことによって，ブランチダイバージェンスが発生した時にコアのアイドルタイムを短くしている．しかし，この手法は分岐先に類似する処理が少ない場合，得られる効果が少なくなってしまう．マルチエージェントシミュレーションでは，エージェント同士の行動が異なることが考えられるため，ブランチダイバージェンスの影響を十分に低減することができない．

また，スレッドデータを並び替えることによるブランチダイバージェ

ンスの低減手法 [14] が提案されている．コンパイル時にプログラムの制御フローを解析し，各スレッドがそれぞれの基本ブロックに入る回数をカウントした配列を作成する．この配列を用いて，スレッドが同様の動作をするように入力データを並び替えることにより，ブランチダイバージェンスの発生を抑制している．この手法では，各スレッドが基本ブロックに入る回数をカウントすることから，コード解析に時間がかかってしまう．また，配列のサイズがスレッド数とプログラムサイズに依存するため，スレッド数とプログラムサイズの増加によって爆発的に増加してしまう．

3.2.2 マルチエージェントシミュレーション

3.2.1 項で述べた手法の他に，エージェントに焦点を当てた高速化手法が提案されている，

エージェントベースのモデリングフレームワークである，Flexible Large-Scale Agent-Based Modeling Environment（FLAME）フレームワークを拡張した FLAME GPU では，エージェントを状態に基づいてグループ化することでブランチダイバージェンスを低減している [15]．ユーザが手動でエージェント関数内の分岐構造を解析し，分岐内で更新される変数を使用する条件節以外を，分岐構造の外側に抽出する．この時，条件節と抽

出部分との間に依存関係がある場合は複数の状態を持つ単一のエージェント関数として，条件節と抽出部分との間に依存関係が無い場合は単一の状態を持つ複数のエージェント関数として新たに関数を定義する．これにより，ブランチダイバージェンスを引き起こすエージェント関数を，ブランチダイバージェンスの少ない新たなエージェント関数に変換している．

また，エージェントの内部状態によってグループ化を行い，ブランチダイバージェンスを低減する手法 [16] が提案されている．この研究では，近傍内の他エージェントの状態によって行動が変化するエージェントモデルに対して，状態数と近傍領域でのグルーピングを行い，特定状態のエージェントのみを計算対象とすることでブランチダイバージェンスを低減している．

これらの手法は，いずれも簡易的なモデルによる実験に留まっており，状態のみしか考慮していないため，状態以外で行動が変化するモデルに対応することができない．しかしながら，3.1.2 項で述べたように，人間を模したエージェントはこれらのモデルより複雑な意思決定を行う．そのため，エージェントの内部状態を考慮するだけではブランチダイバージェンスを低減しきれていない．

4　提案手法

3.1.2 項で述べたように，従来研究では，内部状態・役割・知識・心理状態・類型などを付与することで，より現実に近いエージェントモデルを構築している．その一方で，3.2.2 項で述べたように，複雑な意思決定を行うモデルに関する高速化は十分ではない．

そこで本研究では，マルチエージェントシミュレーションにおける避難シミュレーションに対する高速化手法として，エージェント特性によるグループ化を提案する．エージェント特性とは，エージェントの行動に影響を与える要素を指しており，エージェントの内部状態だけでなく，上で述べたような，役割・知識・心理状態・類型などが挙げられる．さらに，ある類型が特定の状態の場合に行動を起こすような，エージェントの行動に複数の要因が同時に影響を与える場合，エージェント特性を組み合わせてグループ化を行う．

図 4.6 に提案手法の概要を示す．図 4.6 の例では，エージェントがエージェント特性として類型（Type）や状態（State）を持っているものとする．これらをエージェントデータから抽出してキー配列を作成し，キー配列を用いてエージェントデータのソートを行う．このように，エージェント特性に基づいてソートを用いてデータの再配置を行うことで，一つ

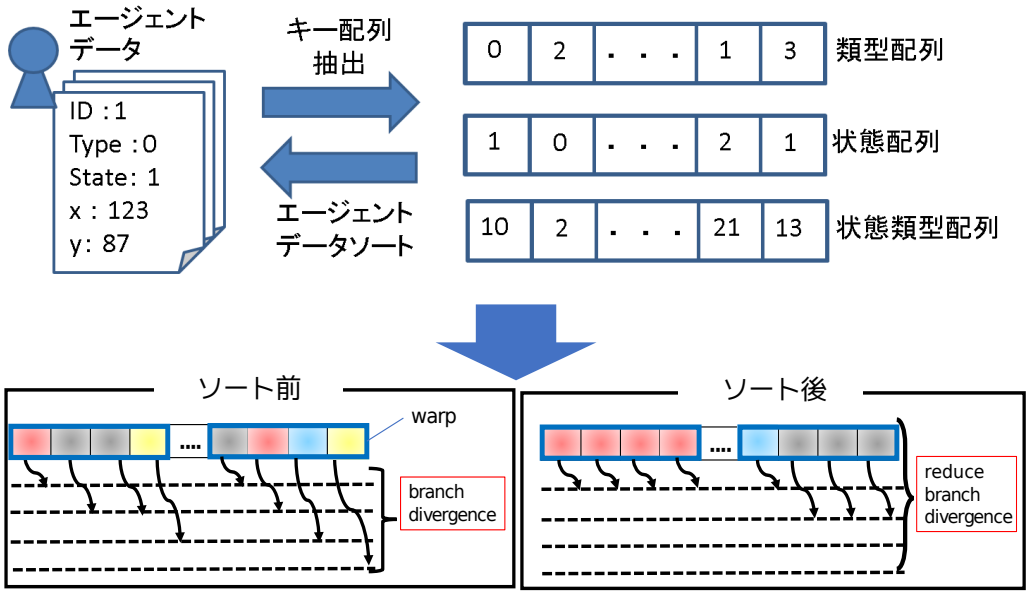

図 4.6: 提案手法の概要

の Warp 内に同様の行動を取るエージェントを配置することができ，ブランチダイバージェンスを低減することができる．また，シミュレーションの進行に伴い，相互作用計算が行われた場合，効果的なエージェント特性は変化すると考えられる．そのため，シミュレーションのステップ数や避難しているエージェントの数といった，シミュレーション環境から得られる情報に基づいて，ソートに用いるエージェント特性をシミュレーション中に変更することで，更なる最適化を行う．

4.1　ソートを用いたデータの再配置

本節では，ソートを用いたデータの再配置について述べる．4.1.1 項ではエージェント特性の選択について述べる．4.1.2 項では複数のエージェント特性の組み合わせについて述べる．4.1.3 項では用いるソートのアルゴリズムについて述べる．

4.1.1　エージェント特性の選択

エージェントデータの配置を最適化するために，分岐を引き起こすようなエージェント特性に基づいてエージェントをグループ化するが，用いるエージェント特性の選択は適切に行われる必要がある．この時用いるエージェント特性は，分岐の条件として用いられるものを選択する．選択すべきエージェント特性はモデルによって異なるため，ブランチダイバージェンスをより低減するためには，どのエージェント特性が構築されたモデルに対して影響が強いかを考慮する必要がある．例えば，ロボカップレスキューのレスキューシミュレーションのように，役割が異なる複数のエージェントが存在するモデルの場合，分岐の条件はエージェントの役割をもとに記述されることが多くなる．このような場合は，エージェントの役割を影響が強いエージェント特性と言えるため，エージェ

ントの役割をキーとして選択する．また，エージェントの内部状態で行動を変化させるモデルの場合は，同様にエージェントの状態を影響が強いエージェント特性と言えるため，エージェントの状態をキーとして選択する．

4.1.2　エージェント特性の組み合わせ

エージェントの行動が現実に近いモデルの場合，一つのエージェント特性をキーとして用いるだけでは，複数のエージェント特性に応じて行動を変化させる場合のブランチダイバージェンスを低減することができない．そのため，複数のエージェント特性を考慮することにより，ブランチダイバージェンスを低減する．例えば，エージェントに心理モデルを採用したモデル [10] では，避難中に他の要因によって行動が変化する．このような場合は，エージェントの内部状態と心理状態の 2 つが影響の強いエージェント特性と言える．また，これら 2 つのエージェント特性を比較すると，内部状態の条件下で心理状態の行動が分岐するため，内部状態の方がより影響の強いエージェント特性と言える．このような場合，内部状態をキーにソートを行った後に心理状態によってソートを行うといった二段階のソートを行う必要がある．この問題を解決するために，影響が強いエージェント特性に重みづけとしてバイアスをかけて桁

を上げ，もう一つのエージェント特性と足し合わせて新たなキーを作成することで，ソート回数の削減を行う．

具体例として図 4.6 のような，類型と状態をエージェント特性に持つ場合を挙げる．さらに，類型よりも状態の方が影響の強いエージェント特性であるとする．状態配列と類型配列を組み合わせる場合，影響が強い状態配列にゲタを履かせ，類型配列と足し合わせることで，新しく状態類型配列を作成している．

4.1.3 ソートアルゴリズム

ソートによるデータの再配置を行う場合，データの一貫性を保つために，エージェントデータとエージェント特性は紐付けてソートを行う必要がある．そこで，CUDA のテンプレートライブラリである Thrust[17] に用意されている sort_by_key() 関数を用いる．この関数は，キーとなる配列とソート対象の配列のインデックスを紐づけ，キーとなる配列を並び替えた際に，対応しているソート対象の配列の要素も同様に並び替える．そのため，エージェントデータの配列とキーとする要素の配列は別に宣言する必要がある．また，ソートによるデータの再配置を行った場合，エージェントデータ配列と他のキー配列の対応付けが崩れないように更新処理を行う必要があるが，マルチエージェントシミュレーション

ではステップ毎にエージェントの更新処理を行うため，大きな問題にはならない．

4.2　考慮するエージェント特性の変更

本節では，シミュレーション中に考慮するエージェント特性を変更する手法について述べる．エージェントの意思決定は，周囲の環境やエージェントから得る情報によって変化する．そのため，4.1 節の手法で用いると決定したエージェント特性は，シミュレーションの進行によって効果的ではなくなってしまう場合がある．そこで，シミュレーションの進行による行動の変化に対応するために，シミュレーションの進行に応じて用いるエージェント特性を変更する．エージェント特性の変更は，シミュレーションのステップ数や避難しているエージェントの数といった，シミュレーション環境から得られる情報によって行う．

4.2.1，4.2.2，4.2.3 項で，シミュレーションのステップ数に応じて，エージェント特性を変更する場合の具体例を示す．例として挙げるモデルは，後述する表 5.2 で示す実験に用いたモデルと同様のものである．このモデルのエージェントは，避難を始める前の未避難状態・避難を行なっている避難状態・避難を完了している避難完了状態の 3 状態を持ち，未避難

状態と避難状態の混雑時にはそれぞれの類型に応じた異なる動作を行う．図 4.7，図 4.8，図 4.9 の上部は避難シミュレーションの状況を示している．また，避難シミュレーション内の緑色の四角は障害物を示し，円はエージェントを示している．図 4.7，図 4.8，図 4.9 の下部は単純な実装と本手法適用後の 0 番目の Warp（Warp 0）と N 番目の Warp（Warp N）内のスレッドとアイドル状況を示している．

4.2.1　シミュレーション初期

図 4.7 にシミュレーション初期の模式図を示す．シミュレーション初期

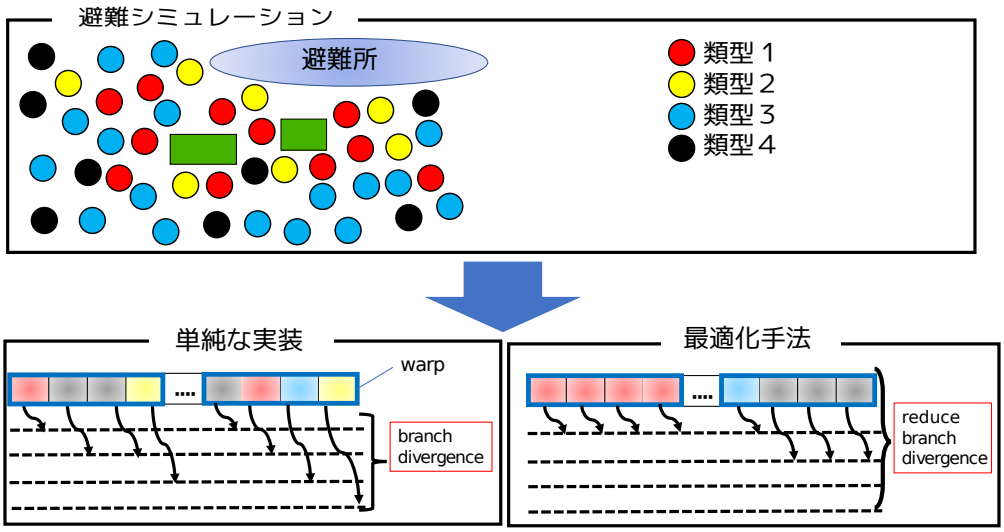

図 4.7: シミュレーションと手法の適用（初期）

では，エージェントは未避難状態であるため，それぞれの類型に応じて異なる動作を行う．よって，単純な実装では Warp 0 に類型 1，2，4 が，

Warp N に類型 1，2，3，4 が混在しているためブランチダイバージェンスの影響が大きい．一方で，最適化手法では類型でソートした結果，Warp 0 は類型 1 のみ，Warp N は類型 3，4 の混在となっている．このため，Warp 0 はブランチダイバージェンスが生じず，Warp N も単純な実装よりブランチダイバージェンスが低減されている．

4.2.2　シミュレーション中期

図 4.8 にシミュレーション中期の模式図を示す．シミュレーション中期

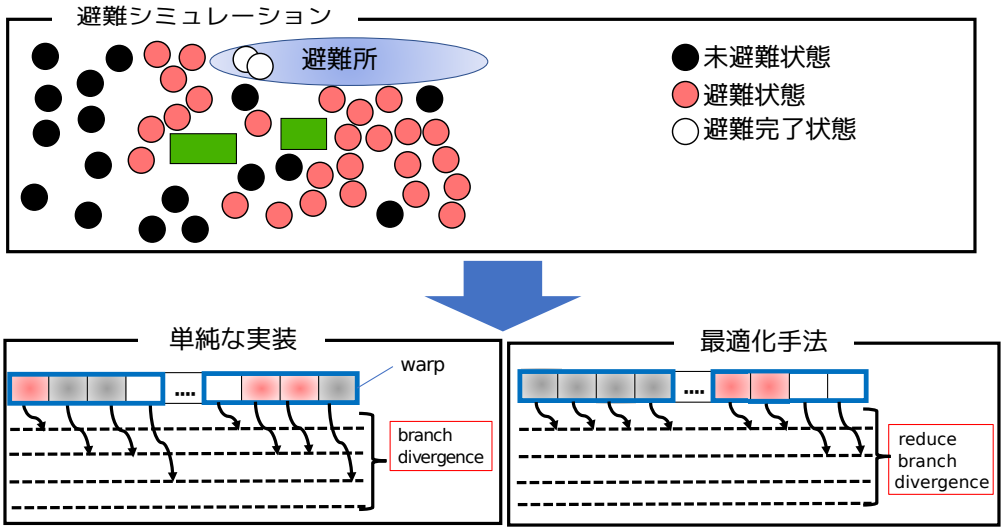

図 4.8: シミュレーションと手法の適用（中期）

では，一部のエージェントを除いて多くのエージェントが避難状態である．この時点では，避難を開始したばかりのエージェントが多く，避難状態のエージェントは順調な避難を行なっており，避難状態のエージェント

行動に類型による差異が見られない．また，避難開始条件が周囲の状況によって左右されてしまう場合，同じ類型内でも状態による行動の差異が発生する．そのため，類型をキーとしたソートのままでは単純な実装と同様にブランチダイバージェンスが発生してしまう．よって，キーを切り替えない単純な実装では Warp 0 と Warp N に 3 状態が混在しているため，ブランチダイバージェンスの影響が大きい．一方で，状態をキーとしたソートへ切り替えた結果，Warp 0 は未避難状態のみ，Warp N は避難状態，避難完了状態の混在となっている．このため，Warp 0 はブランチダイバージェンスが生じず，Warp N も単純な実装よりブランチダイバージェンスが低減されている．

4.2.3　シミュレーション後期

図 4.9 にシミュレーション後期の模式図を示す．シミュレーション後期では, 全てのエージェントが避難を開始しており，避難状態もしくは避難完了状態である．この時点では，多くのエージェントが避難所の近くに殺到しており，混雑によって順調な避難が出来ていない．避難状態のエージェントは，混雑を感じた場合に類型に応じた異なる動作を行う．そのため，状態をキーとしたソートでは状態内の類型の違いにより，単純な実装と同様にブランチダイバージェンスが発生してしまう．よって，キー

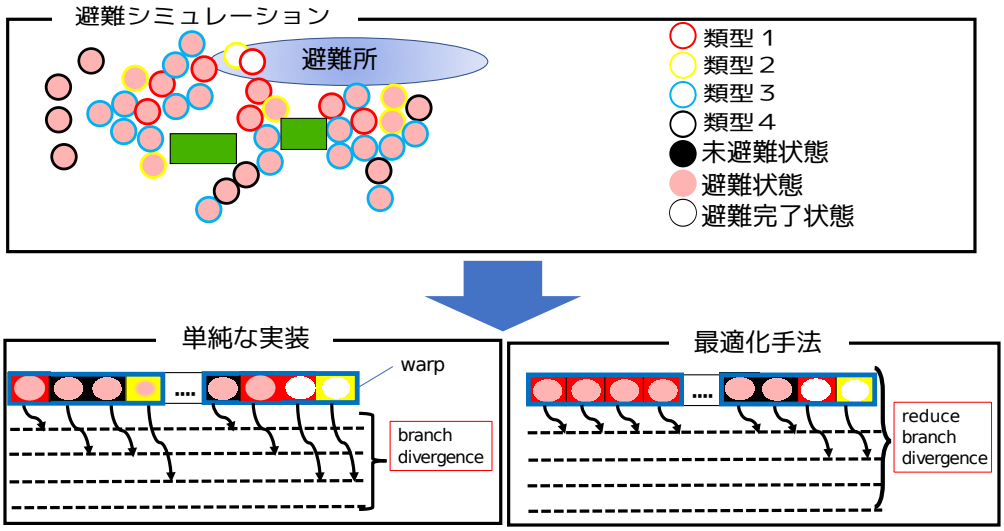

図 4.9: シミュレーションと手法の適用（後期）

を切り替えない単純な実装では，Warp 0 は避難状態の類型 1，2，4 が，Warp N は避難状態の類型 1，2 と避難完了状態が混在しているためブランチダイバージェンスの影響が大きい．一方で，状態と類型を組み合わせて考慮した状態類型ソートに切り替えた結果，Warp 0 は避難状態の類型 1 のみ，Warp N は避難状態の類型 4 と避難完了状態の混在となっている．このため，Warp 0 はブランチダイバージェンスが生じず，Warp N も単純な実装よりブランチダイバージェンスが低減されている．

5　評価

避難シミュレーションに提案手法を実装し，様々な条件でシミュレーションを行い，提案手法によるシミュレーション結果と状態によるグループ化の結果を比較した．5.1 節では，実行環境とエージェントのモデル，実験のテストケースについて述べる．5.2 節では，評価結果について述べる．

5.1　実行環境と実験内容

避難者エージェントが避難所へ向かうシミュレーションプログラムに提案手法を実装し，シミュレーションにかかった時間を評価した．評価に用いた環境は表 5.1 の通りである．

表 5.1: 評価環境

	環境
CPU	Intel(R) Core(TM) i7 CPU 930
Memory	6GB
GPU	GeForce GTX TITAN
Memory	6GB

エージェントは状態と類型を持ち，状態・類型・状態と類型によって行動が変化するモデルを用いた．今回キーとして選択したエージェント特性は，状態，類型，状態と類型の三つである．また，手法のオーバヘッドを確認するために，ソートによるグループ化を行わない場合の，シミュ

レーションにかかった時間も測定した．さらに，シミュレーション中に類型，状態，状態と類型の順に，キーとするエージェント特性を切り替えた場合の，シミュレーションにかかった時間も測定した．エージェントの局所的な行動は Sim Tread[7] を参考に実装した．シミュレーションプログラム中の類型と類型による行動は関連研究 [11] を参考に，状態による行動の差を与えて実装した．それぞれの状態による他類型との行動の比較は，表 5.2 の通りである．

表 5.2: エージェントの行動

他類型との比較			
未避難状態	通常避難状態	混雑時	避難完了状態
異なる	一致	異なる	一致

今回は手法の有用性を示すために，エージェント数，類型の種類数，類型間でのエージェント数の内訳を変更し，実験を行った．それぞれのテストケースの概要を表 5.3 と表 5.4 に示す．類型数 16 の場合，他類型との差異は表 5.2 と同様で，未避難状態と混雑時に他類型と異なる行動をとる．また全ての条件において，シミュレーションを 3 回ずつ行い，その平均を実験結果とした．

表 5.3: 本実験に用いたテストケース

テストケース	ステップ数	エージェント数	類型数
1:基本	2,100	2^9〜2^{19}	4
2:類型間の偏り	2,100	2^9〜2^{19}	4
3:類型の種類増加	2,100	2^9〜2^{19}	16

表 5.4: 本実験に用いたテストケース続き

テストケース	類型の内訳
1:基本	類型 A:20 %，類型 B:20 %，類型 C:40 %，類型 D:20 %
2:類型間の偏り	類型 A:70 %，類型 B:15 %，類型 C:10 %，類型 D:5 %
3:類型の種類増加	全類型均等に割り振り

5.2 評価結果

状態を用いる手法とソートを行わない場合に対し，類型を用いる手法・状態と類型を組み合わせる手法・用いるキーを切り替える手法の性能評価を行った．これ以降，状態を用いる手法を状態ソート，ソートを行わない場合をソート無し，類型を用いる手法を類型ソート，状態と類型を組み合わせる手法を状態類型ソート，用いるキーを切り替える手法をキー切り替えと呼ぶこととする．

図 5.10，図 5.11，図 5.12 は，状態ソートの実行時間を 100%としたときの実行時間比を表したグラフである．縦軸は状態ソートに対する実行時間比を表しており，横軸はエージェント数を表している．またそれぞれ

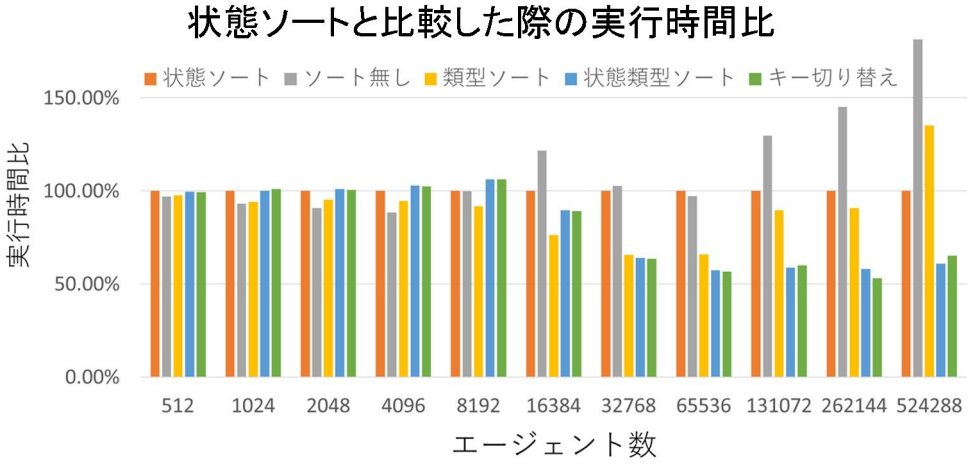

図 5.10: テストケース 1 についての評価結果

の棒グラフは左から，状態ソート・ソート無し・類型ソート・状態類型ソート・キー切り替えを表している．

図 5.10 はテストケース 1 の結果を示している．4096 エージェントまでは，状態ソートよりも類型ソートの方が実行時間の削減ができているが，ソート無しと比べると実行時間が長いことから，手法による効果よりもオーバヘッドの方が大きいことがわかる．一方エージェント数が 16384 以上の時，状態ソートとソート無しよりも，状態類型ソートとキー切り替えの方が，シミュレーション時間が短くなっており，性能向上が確認できる．

図 5.11 はテストケース 2 の結果を示している．8192 エージェントまで

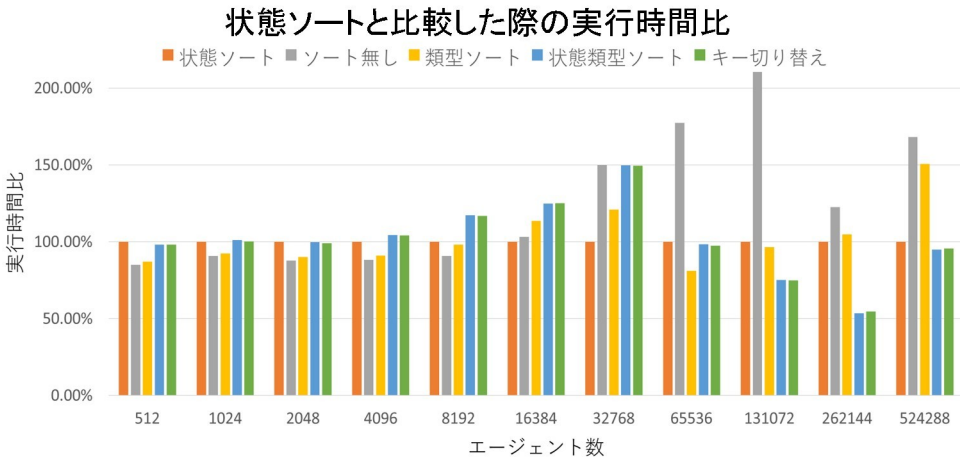

図 5.11: テストケース 2 についての評価結果

は，状態ソートよりも類型ソートの方が実行時間の削減ができているが，ソート無しと比べると実行時間が長いことから，手法による効果よりもオーバヘッドの方が大きいことがわかる．また，16384 エージェントと 32768 エージェントでは，状態ソートの方が提案手法と比べてシミュレーション時間が短いため，状態ソートが提案手法より優位であることが確認できる．一方エージェント数が 65536 以上の時，状態ソートとソート無しよりも，状態類型ソートとキー切り替えの方が，シミュレーション時間が短くなっており，性能向上が確認できる．

図 5.12 はテストケース 3 の結果を示している．2048 エージェントと 4096 エージェントでは，状態ソートよりも類型ソートの方が実行時間の削減ができているが，ソート無しと比べると実行時間が長いことから，手

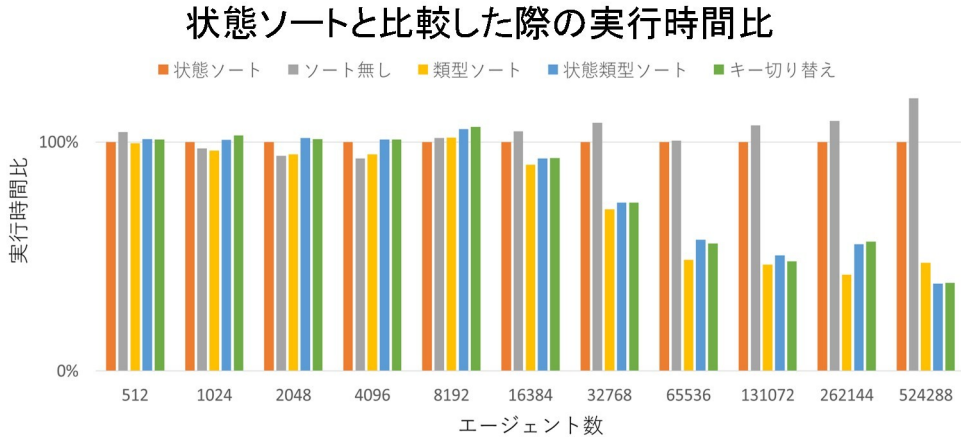

図 5.12: テストケース 3 についての評価結果

法による効果よりもオーバヘッドの方が大きいことがわかる．8192 エージェントでは，状態ソートの方が提案手法と比べてシミュレーション時間が短いため，状態ソートが提案手法より優位であることが確認できる．一方エージェント数が 16384 以上の時，状態ソートとソート無しよりも，状態類型ソートとキー切り替えの方が，シミュレーション時間が短くなっており，性能向上が確認できる．

6　考察

本章では，5.2 節で示した評価結果に対する考察を述べる．

全てのテストケースにおいて，エージェント数が少ない場合に，類型ソート以外では性能向上が見られなかった．これは，ソートによるオー

バヘッドが，ブランチダイバージェンスの削減による性能向上よりも大きかったためと考えられる．類型はシミュレーション中に変更されない静的なエージェント特性であり，ソートを行う回数が最初の一回だけで良いことから他の手法と比べてオーバヘッドが小さい．そのためエージェント数が少ない場合には，類型を用いる手法でのみ性能向上が確認できたと考えられる．

また，テストケース 1 とテストケース 2 について，エージェント数が増加するほど類型ソートの性能が低下してしまうことが示された．これは，シミュレーションの進行に伴い，類型内で状態によってエージェントの行動に差が出ることによって生じるブランチダイバージェンスが原因であると考えられる．エージェント数が増えることによって，このようなブランチダイバージェンスの発生回数が増え，性能低下に繋がってしまっている．同様に，状態ソートでも状態内での類型の差によって，ブランチダイバージェンスによる性能低下が発生していることが考えられる．その一方で，状態類型ソートとキー切り替えは，状態ソートや類型ソートと比べると実行時間が短い．よって，状態類型ソートとキー切り替えは，状態や類型のみを考慮しただけでは低減しきれないブランチダイバージェンスを低減することができたと言える．

テストケース 1 に比べると，テストケース 2 の場合の性能向上が小さく，テストケース 3 の場合の性能向上が大きいことが示されている．テストケース 2 については，類型間のエージェント数の内訳に偏りがあるテストケースであるため，一つの類型に属するエージェントが多く存在している．そのため，ランダムなデータ配置をした場合であっても，同じ類型が同じ Warp に割り当てられることが多く，類型によるブランチダイバージェンスが発生する可能性が低いと考えられる．そのため，類型ソートによる効果が薄く，類型ソートの性能向上が小さかったと考えられる．テストケース 3 に関しては類型の種類が多いテストケースであるため，状態ソートでは同じ Warp 内に複数種類の類型が入ってしまう可能性が高いと考えられる．この場合，同じ Warp 内に割り当てられた全ての類型による処理を逐次的に行なってしまうため，ブランチダイバージェンスによる性能低下が非常に大きくなってしまうと考えられる．

以上より，エージェント数が多い場合に，エージェント特性の適切な選択によって，提案手法はブランチダイバージェンスを低減することができ，従来手法よりシミュレーション時間の削減を達成できたといえる．

7　まとめと今後の課題

本論文では，GPU 上のより現実に近いモデルを採用するマルチエージェントシミュレーションに対して，従来の内部状態のみではなく，エージェントの行動決定に関わる要素であるエージェント特性によるグループ化を提案した．さらに，複数のエージェント特性を組み合わせ，用いるエージェント特性をシミュレーション中に切り替えることにより，ブランチダイバージェンスを低減し，最大で 62%の高速化を実現した．

今後の課題として，シミュレーション中でエージェント特性を切り替える条件の最適化と自動的なエージェント特性の選択が挙げられる．これらは，シミュレーションシナリオに応じて変化するため，最適化にはモデルに対する深い理解が必要であり，現状ではどちらも最適化できていない．また，エージェント数が少ないシミュレーションで本手法を適用する場合は，オーバヘッドの低減も課題として挙げられる．

謝辞

本研究を行うにあたり，多数のご指導をいただきました大野和彦講師，高木一義教授，並びに深澤祐樹技術職員に深く感謝いたします．また，コンピュータアーキテクチャ研究室の学生には常に刺激的な議論を頂き，精

神的にも支えられました．合わせて感謝をいたします．

参考文献

[1] J. Tsai et al., ”ESCAPES - Evacuation Simulation with Children, Authorities, Parents, Emotions, and Social comparison”, International Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp.457-464, 2011

[2] Yuksel M.E. , ”Agent-based evacuation modeling with multiple exits using NeuroEvolution of Augmenting Topologies”, Advanced Engineering Informatics, Vol.35, pp.30-55, 2018

[3] CUDA C programming-guide, `https://docs.nvidia.com/cuda/cuda-c-programming-guide/`

[4] Macal C. M., North M. J., ”Tutorial on agent-based modelling and simulation”, Proceeding of the Winter Simuation Conference, pp.2-15, 2005

[5] B.Lenzitti, D.Tegolo, C.Valenti, "Prey-predator strategies in a multiagent system", Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05), pp.184-189, 2005

[6] Helbing D, Molnar P., "Social force model for pedestrian dynamics", Physical Review, E 51, pp.42824286, 1995

[7] 木村 謙, 佐野 友紀, 林田 和人, 竹市 尚広, 峯岸 良和, 吉田 克之, 渡辺 仁史, マルチエージェントモデルによる群衆歩行性状の表現―歩行者シミュレーションシステム SimTread の構築―, 日本建築学会計画系論文集, 74 巻, 636 号, pp.371-377, 2009

[8] レスキューシミュレーションリーグ, http://www.robocup.or.jp/robocup-rescue/simulation/, 2020 年 1 月 30 日アクセス

[9] 松島 弘, アランニャ・クラウス, 狩野 均, 避難者の情報伝達を考慮した地震・津波避難シミュレーションのためのマルチエージェントモデル, 情報処理学会研究報告, 2015MPS106 巻, 4 号, pp.16, 2015

[10] 近田 洋輔, 原山 美知子, 被災者の心理に基づく津波避難シミュレーション, 情報処理学会研究報告, 2013-IS-126 巻, 8 号, pp.1-8, 2013

[11] 上田　遼, 行動特性に着目した津波避難の分析と対策―人間と ICT の相互作用による安全避難の共創―, 日本地震工学会論文集, 17 巻, 4 号, pp.140-169, 2017

[12] サーベイリサーチセンター, 宮城県沿岸部における被災地アンケート, https://www.surece.co.jp/wp_surece/wp-content/uploads/2017/10/20110311_miyagi.pdf, 2011

[13] Tianyi Dvid Han, Tarek S. Abdelrahman, "Reducing branch divergence in GPU programs", Proceeding of the Fourth Workshop on General Purpose Processing on Graphics Processing Units, Article No.3, pp.1-8, 2011

[14] Zheng Cui, Yun Liang, Kyle Rupnow, "An Accurate GPU Performance Model for Effective Control Flow Divergence Optimization", Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium(IPDPS), 2012

[15] Mozhgan K. Chimeh, et al., "Simulating heterogeneous behaviours in complex systems on GPUs" Simulation Modelling Practice and Theory, Vol.83 pp.3-17, 2018

[16] 廣田　裕哉, 状態を用いたマルチエージェントシミュレーションのスケジューリング最適化, https://mie-u.repo.nii.ac.jp/?action=pages_view_main&active_action=repository_view_main_item_detail&item_id=13031&item_no=1&page_id=13&block_id=21, 2018

[17] Thrust, https://docs.nvidia.com/cuda/thrust/index.html

A　プログラムリスト

B　評価用データ