4F1-5in

災害などの大規模マルチエージェントシミュレーション実現に向けたスケーラビリティ改善のためのフレームワークの試作

A Preliminary Framework to Improve the Scalability of Large-scale Multi Agent Simulations

佐野 義仁[*1] 福田 直樹[*1]
Sano Yoshihito Fukuta Naoki

[*1]静岡大学大学院情報学研究科
Graduate School of Informatics, Shizuoka University

Understanding the movements of people, vehicles, and other moving objects is demanded for better analysis of disasters or events occurred. Multi-agent simulation is often used for such analysis. In order to improve the reproducibility of real situations, agents should respond to dynamic environmental changes, as well as considering efficient computation of them since the simulation often becomes huge scale. In this paper, a GPGPU-based efficient and scalable framework is presented, by applying OpenCL. Furthermore, we present a platform to easily test and try the implemented path planning codes in various settings.

1. はじめに

マルチエージェントシミュレーションはすでに様々な対象に対して適用されており，例えば道路交通シミュレーション [1]，群衆シミュレーション [7]，また空港の避難シミュレーション [5] などへの適用が行われている．

このようなエージェントシミュレーションにおいては，シミュレーションの詳細度をあげることが重要な課題の 1 つである．例えば，空港の避難シミュレーションにエージェントシミュレーションを導入した例 [5] では，今まであまり考慮されてこなかった人間の恐怖といった感情や家族といった人間関係をエージェントシミュレーションに導入して避難シミュレーションを行なうことで，今まで正常な避難ができると考えられていた災害シナリオにおいて，逃げることができない人間 (エージェント) が存在する可能性を示している．災害やイベントなどの非日常的な状況に対して，車両や人々がどのように行動をするのか分析したり，あるいはそこで生じる現象や問題に対して精密な議論に基づいて対策を考えたりする場合には，エージェントが動的な状況の変化に適切に反応して行動できるようにプログラムされていることが必要となる．すなわち，エージェントが動的な環境変化に対応できるようにコーディングされ，それを合理的な時間内でシミュレーションにより動作させることができる必要がある．動的な環境変化に対応するというシナリオとしては，例えば，道路交通の分野では，シミュレーション実行途中での細粒度なエージェントの再プラニング処理が提案されている [2].

エージェントシミュレーションでは，シミュレーションの大規模化も 1 つの課題である．現実の世界では，対象を 1 つの都市に注目して考えたとしても，その 1 つの都市の交通が他の都市における交通流入・流出量に影響される場合もあるほか，たとえ 1 つの都市をシミュレートするためにも，何百万台の車両が存在する状況を再現できることが重要となる．例えば，電気自動車専用レーンを都市に導入したらどのような影響が出るかということを検討するためマルチエージェントシミュレーションが利用されたが，この際には，およそ 300 万個のエージェントによるシミュレーションが行われた [3].

連絡先: 佐野義仁, 静岡大学大学院情報学研究科, 〒 432-8011, 静岡県浜松市中区城北 3-5-1，gs13017@s.inf.shizuoka.ac.jp

マルチエージェントシミュレーションでは大規模化と処理の効率化が 1 つの課題である [8] が，エージェントが状況に合わせて動的に経路探索を行おうとすると処理が複雑になり [2]，シミュレーションの大規模化が容易ではない．

本研究では，動的な環境変化にエージェントが対応できるように実装されたコードの処理を高速化することによって処理の負荷を減らし，マルチエージェントシミュレーションのスケーラビリティの改善を目指す．具体的には，エージェントシミュレーションの処理を GPGPU などの計算資源を有効活用することで効率的に処理可能とするためのフレームワークを用意し，スケーラビリティの改善を容易にする．

2. 本研究の位置づけ

シミュレーションのスケーラビリティを改善させるための方法として，Hadoop やクラウドといった技術を利用して複数のコンピュータに処理を行わせる方法が考えられる．複数のコンピュータを利用して処理を分散させることによって全体としての処理の高速化ができ，結果としてスケーラビリティの改善につなげることが可能である．しかしながら，シミュレーションの状況に対するデータをコンピュータ間で同期させる必要があり，そのデータの同期をするために膨大な通信を行い，効率的にスケーラビリティを高めることは容易ではない．またシミュレーションを行いたい人が常に複数のコンピュータを用意することができるとは限らない．このような点を考慮して，複数のコンピュータを利用したシミュレーションの大規模化を検討する前段階として，本研究では 1 台のコンピュータに対して注目し，シミュレーションの大規模化について検討をしていく．1 台のコンピュータの処理について注目したとき，主に CPU(Central Processing Unit) を利用して多くの処理は行なわれてる．しかしながら，グラフィック処理用に高度な演算性能を持つ，GPU(Graphics Processing Unit) が用意されている．この GPU のリソースを有効活用するために，GPGPU という技術によって，GPU 上で汎用的な演算をエージェントシミュレーションに対しての適用を容易にすることにより，高速に処理を行うエージェント内部処理のコードを容易に開発することができると考えられる．そこで，本研究では，エージェントシミュレーションにおける処理を GPU によって行う際に，それらのコーディング，検証，パラメータチューニング等を効

果的に支援し，その成果物を容易に実シミュレーションシステムに組み込めるようなフレームワークの構築を考える．

3. 提案フレームワークの概要

GPGPU を行うためのフレームワークには，NVIDIA が提供する CUDA， ATI が提供する ATI Stream，また，OpenCL といったものがある．本研究では，シミュレーションを行いたい人が様々な会社の GPU を利用する可能性，またマルチ CPU といったデバイスに処理を行わせることなどを考え，CPU や GPU，Cell プロセッサなど様々なものをデバイスとして利用することが可能である OpenCL を利用する．

また，エージェントシミュレーションを利用することを考えた際，利用したい人がコンピュータの専門家であるとは限らない．そこで本研究では，災害などのシミュレーションを行いたい利用者に負担があまりなく (OpenCL などを利用して)GPGPU を行うことができるようなシステムを作成する．

本研究では，エージェントが，動的な環境変化に対応をするために，エージェントがシミュレーションの途中で目的地までのルートの再決定などをする再プランニングを行うことを想定する．そこで，本研究では，プランニング処理を GPGPU を用いて計算を行えるようにする．GPU は， 同時に同じ処理を行うことを得意とする．すなわち 1 つのカーネルプログラムを並列的に実行することに向いている．したがってプランニングを行う際に同じ動作をする箇所をカーネルプログラムとして記述し，カーネルプログラムを複数呼び出すことによって実現する方法が有効であると考えられる．また，エージェントシミュレーションでの経路探索処理について考えたとき，エージェントそれぞれの個性を表現するためにそれぞれのエージェントに対して適用する経路探索アルゴリズムが異なる場合があるということや，シミュレーションを行う人によって適用したいアルゴリズムが違うということを考慮し，本研究では，経路探索を行うアルゴリズム自身の高速化とは別のアプローチをとり，経路探索処理自体の高速化を目標として，エージェント毎のプランニング処理を GPU を用いて並列的に行うような方法を取る．

OpenCL では，データ並列とタスク並列という 2 つの方法を用いて並列的に処理をすることができる．そのうち，データ並列では，複数のデータに対して，同じ処理 (カーネル) を各プロセッサで並列的に処理を行う．複数のデータを一度に処理をすることによって高速化が可能である． GPU のような多くのプロセッサを持っているデバイスを利用してデータ並列を行うことは有効な手段である．そこで本研究では，エージェント毎の経路探索処理をデータ並列によって並列化を行い，高速化を行う．

本研究で試作をするフレームワークを利用するユーザは，OpenCL を C 言語上で用いて，自分が利用したい経路探索アルゴリズムを記述する．この際，道路網のデータやエージェント数といったデータを引数として受け取ることができ，ユーザは，それらの引数を利用しつつ記述をしていく．記述をしてプログラムが完成したらユーザは，作成した関数名を指定して登録をする．このようにすることによってユーザはシミュレーション上で自分の指定した経路探索アルゴリズムを OpenCL を用いて実行することが可能である．また，フレームワークを通して，MATSim といったシミュレーションと連結を取ることが可能にするための機能の実装も行う．本研究で作成するフレームワークを通すことによって，MATSim 上で行うプランニング処理を OpenCL を用いて行えるようにするために必要な処理が行われる．この際，MATSim などのシミュレーションでは，Java などの言語でプログラムが記述されている可能性があり，直接 OpenCL を用いたプランニングを行うことができない場合がある．そこで，本研究では，OpenCL C 言語で記述された経路探索プログラムと OpenCL を直接利用できないプログラミング言語との仲介をする．具体的には，本研究では，Java 言語で記述されたシミュレーションプラットフォームで利用できるようにするために，OpenCL の Java で利用するためのライブラリで利用できる形に変換することによってこの機能を実現する．

本研究で提案するフレームワークでは，道路網を 2 通りの方法で取得することができるようにする．1 つ目の方法として，利用するユーザが，手動で道路網を作成する方法である．本研究で提案するフレームワークのマップデータ作成機能を用いて，本システムの画面上にノードを配置し，ノード間をリンクで接続していくことによって地図データを作成することを可能にする．このように作成した地図データは，保存をすることが可能であり，保存した地図データを再び読み込んでシミュレーションに再利用をしたり，地図を編集することによって新しい地図を作成したりすることができる．2 つ目の方法として，道路交通シミュレーションのプラットフォームである MATSim で用いられている地図データを利用することができるようにする．MATSim で用いられている地図データは，XML 形式で記述されており，XML 形式で記述されたノード情報とリンク情報を読み込み，それぞれに関連したパラメータの値を取得することによって，道路網を作成する．これによって MATSim で用いられている地図データをそのまま流用して，本研究で提案するフレームワークで用いることが可能である．また，作成された道路網に対して，新たにノードを配置し，リンクで接続することによって道路網を拡張することが可能である．このような方法をとることによって，地図データを利用するユーザ自身で，常に最新の道路網にすることができる．

本研究の目的は，マルチエージェントシミュレーションのスケーラビリティの改善である．しかしながら，大規模な道路網を用いて，シミュレーションを行おうとすると，用いる経路探索アルゴリズムによっては，1 つの GPU のコアに行わせる処理が膨大なものとなってしまい，正しく処理を行うことができない可能性が考えられる．そこで，本研究では，どの程度の大きさならば GPU 上で処理を行うことができるか検討するための機能として，道路網を一時的に縮小する機能を作成する．この機能では，道路網状のリンクをいくつか切断することによって，道路網の規模を小さくする，この際，リンクを切断することからノード間のつながりがなくなり，あるノード間で行き来ができなくなる可能性が現れる．すなわち，このような状態になると，経路探索アルゴリズムを用いて経路を求めることができなく可能性がある．そこで，このようなことを防ぐために，すべてのノードからリンクによって接続されるノードを作成する．このような方法によって，道路網の規模を小さくしつつ，OpenCL に対するテストなどを行うことができる．

本研究では，道路交通を対象として災害やイベントなどが発生する大規模シミュレーションを行うことを目標としている．災害やイベントが発生した際のことを考えると，通常の道路網の状態が保たれるのは困難であり，道路の寸断や通行止めになる可能性が考えられる．そこで，このようなことを想定し，本研究では，道路網に対して変化を与えるための機能を実装する．この機能では，ノード同士をつなげているリンクを切断することによって道路をエージェントが通行できないようにする．この際，例えば震災などが発生した場合は長期間道路が寸

断されている可能性があることや，車などの事故による通行止めでは比較的短期間の通行止めですむ可能性が高いといったようなことを考慮し，道路の変化時間を任意に決められるようにする．このようにすることによって，道路網が通行止めになるような様々なイベントに対応することが可能となる．

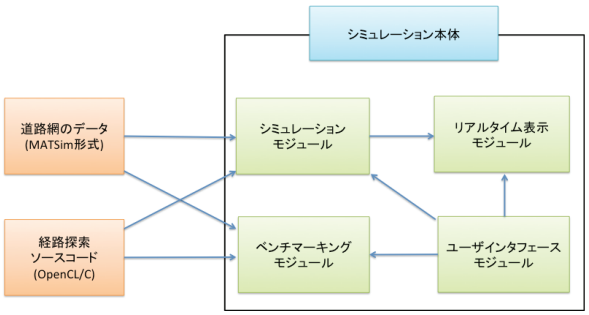

図 1: 提案フレームワークの概要

4. フレームワークに基づいた実装環境

現在，本研究で提案するフレームワークに基づいて実装環境の制作を行っており，OpenCL を用いて処理を行うことができる実装環境を実際に作成した．図 2 は作成した実装環境である．この実装環境では，簡易的な道路交通シミュレーションを行うことが可能であり，またエージェントのプランニング処理を，OpenCL を用いて行うことが可能である．実際に本研究では，Dijkstra 法，A*法，RTA*法 [4]，LRTA*法 [6] の 4 つの経路探索アルゴリズムに対して，OpenCL の適用を行い，OpenCL を用いたプランニングを行えるようにした．

この作成した実装環境を用いて，プランニング処理に対する並列処理についてのテストを行った．プランニング処理のテストでは，それぞれエージェントに出発地と目的地を与え，すべてのエージェントが出発地から目的地までの経路を求めるためにかかる処理時間を計測した．テストにはノード 12，リンク 22 のマップを用い，実験環境としては，OS: OS X 10．8．2，CPU：2．4 GHz Intel Core 2 Duo，コンパイラ: gcc4．2．1 build 5658，GPU: NVIDIA GeForce 320M，メモリ: 8 GB 1067 MHz DDR3 の MacBook Pro を使用した．

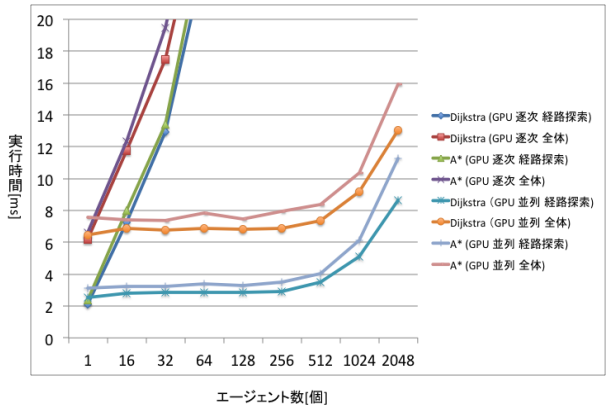

図 3: 並列処理に関するテスト

図 3 は，Dijkstra 法，A*法に対して，OpenCL を用いて GPU で処理を行った結果である．横軸をエージェント数，縦軸を処理時間とした．またグラフ項目である「GPU 逐次 経路探索」は，1 つの GPU コアですべてのエージェントを逐次的に処理をした際に経路探索にかかる時間を表し，「GPU 逐次 全体」は，逐次的に処理をした場合で，経路探索にかかる処理時間にさらに OpenCL を利用するために必要な処理にかかる時間を足した合計の時間である．同様に，「GPU 並列 経路探索」は，複数の GPU コアで並列的に処理をした際に経路探索にかかる時間を表し，「GPU 並列 全体」は，並列的に処理をした場合で，経路探索にかかる処理時間にさらに OpenCL を利用するために必要な処理にかかる時間を足した合計の時間である．図 2 の結果から，それぞれの経路探索アルゴリズムにおいて，逐次的に実行した場合はエージェント数が増えるにつれ，それに比例して処理時間が増えるが，並列的に処理をした場合，実験環境においては，エージェント数 256 まではほぼ処理時間が変わらないことを観測することができた．この結果から，多数ある GPU コアを用いて並列的にプランニング処理を実行できていることが確認できた．また，RTA*法と LRTA*法に対しても同様なテストを行い，同様なプランニング処理の並列動作を確認することができた．

次に，利用する GPU によって，どの程度性能差が生まれるのかについて検証が実際に本フレームワークを用いて行えることを，次の手順により確かめた．比較する GPU として，先の実験で用いた NVIDIA GeForce 320M，同じ NVIDIA 製の NVIDIA GeForce 8800GT，また AMD 製の Radeon HD 6750M を利用する．

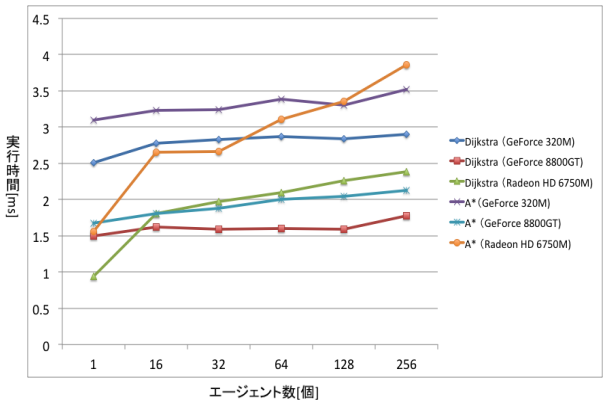

図 4: 複数の GPU による比較

図 4 は，Dijkstra 法と A*法に対して比較をした結果である．横軸をエージェント数，縦軸を処理時間とした．グラフ項目は，それぞれの GPU で Dijkstra 法と A*法を処理したものとなる．まず，同じメーカーの GPU である GeForce 320M と GeForce 8800GT では，GeForce 8800GT の方が，少ない時間で処理を行うことができることを観測することができた．これは，同種の GPU に対して，GeForce 320M と GeForce 8800GT の処理能力の差が本フレームワーク上でも確認できたということである．また，図 3 の結果では，AMD 製の Radeon HD 6750M は，エージェント数が少ないときには，NVIDIA 製のものよりも処理時間が短く，エージェント数が多くなると逆転し，NVIDIA 製のものよりも処理に時間がかかってしまうという現象が観測された．すなわち，NVIDIA 製の GPU では，AMD 製のものに比べてエージェント数が増えても処理時間があまり変わらなく，AMD 製の GPU は，NVIDIA 製のものに比べて並列的に動作させた際のオーバヘッドが大きく生じるという状況を，本フレームワークを用いても正常に再現でき

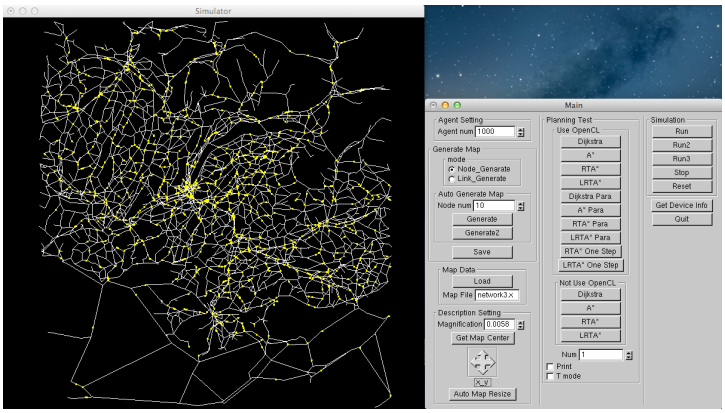

図 2: 実装環境の概観

た．この結果から，AMD 製の GPU に向けた動作では，相応のチューニングを行わないと，GPU の性能を有効に使えない可能性が示唆される．今回は，プログラム上のメモリ配置や最大並列度等を示すパラメータをすべて共通の状態で動作検証を行っている．それぞれの GPU に最適化されたパラメータに自動的に合わせられるような仕組みの実現に向けて，手法の検討と本フレームワークへの実装を現在進めている．

また，多様な GPU に対する性質の違いを，複数の実行環境間で一括して動作試験・調整することを可能にするための機構についても検討しており，その実装は今後の課題となる．

5. まとめ

本研究では，災害やイベントなどが発生したときの人々や車両の動きを理解するための方法として，マルチエージェントシミュレーションを利用することを考え，スケーラビリティ改善に役立つフレームワークの試作を行った．本研究では，その適用対象を，道路状況の変化を考慮したマルチエージェント交通シミュレーションとした．具体的には，災害の発生による道路網の変化や現在の道路網の渋滞状況などを考慮してエージェントがシミュレーションの途中に再プランニングをしながら行動をしていく部分への適用を試みた．本研究で作成したフレームワークを用いることによって，エージェントの行動を決定するプランニングなどの処理を OpenCL を用いてコーディングした際に，その実行時の特性やパラメータチューニング等の作業における負荷の低減が期待される．

今後の課題としては，本研究で作成したフレームワークによってどの程度スケーラビリティの改善に寄与することが可能であるかを，具体的なシミュレーション問題における改善事例などを通じて評価することがある．また，処理の高速化として，複数の GPU を組み合わせるような場面への適用を容易にすることも考えられる．複数の GPU を用いる方法としては，1 台の計算機に複数の GPU を搭載する方法と，GPU を搭載した計算機を複数用いる方法，または両方を組み合わせた方法などが考えられる．このような環境では，あらゆる環境の組み合わせを実環境として準備することが容易ではないため，ある程度の典型的な構成の機材とそれらから事前に収集した実行特性を加味して，効果的な環境を実現する機材の組み合わせを，それらを準備することなく事前に検証できるようなフレームワークの実現も考えたい．現段階では，テストを行いたい環境でプログラムを実行し，手動でテストを行うことによって，GPU に対してのプランニング処理にかかる時間を取得している．典型的な機材構成となるテスト環境を用意したコンピュータ同士をネットワーク上で接続し，検証結果をまとめることは，比較的容易である．そこから種々の機材構成応じた演算性能を予測するためのパラメータ抽出に関しての検討を現在進めている．

参考文献

[1] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K. Axhausen. Agent-based simulation of travel demand: Structure and computational performance of matsim-t. In *Proc. the 2nd TRB Conference on Innovations in Travel Modeling*, 2008.

[2] E. de la Hoz, I. Marsa-Maestre, M. A. Lopez-Carmona, and P. Perez. Extending matsim to allow the simulation of route coordination mechanisms. In *Proc. The 1st International Workshop on Multi-Agent Smart Computing(MASmart 2011)*, pages 1–15, 2011.

[3] R. Kanamori, T. Morikawa, and T. Ito. Evaluation of special lanes as incentive policies for promoting electric vehicles. In *Proc. The 1st International Workshop on Multi-Agent Smart Computing(MASmart 2011)*, pages 45–56, 2011.

[4] R. E. Korf. Real-time heuristic search. In *Artificial Intelligence*, pages 189–211, 1990.

[5] J. Tsai, N. Fridman, E. Bowring, M. Brown, S. Epstein, G. Kaminka, S. Marsella, A. Ogden, I. Rika, A. Sheel, M. E. Taylor, X. Wang, A. Zilka, and M. Tambe. Escapes - evacuation simulation with children, authorities, parents, emotions, and social comparison. In *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS2011)*, pages 457–464, 2011.

[6] 石田 亨, 新保 仁. 実時間探索による経路学習. 人工知能学会誌, 11(3):411–419, 1996.

[7] 山下 倫央, 岡田 崇, 野田 五十樹. 大規模群集流動の制御に向けたシミュレーション環境の構築. In *Joint Agent Workshop and Symposium(JAWS)*, 2012.

[8] 中島 悠, 山根 昇平, 服部 宏充. マルチモデルに基づく都市交通プラットフォーム In *Joint Agent Workshop and Symposium(JAWS)*, 2010.