柔軟に構成を変更可能な 人流・交通流シミュレータの設計と評価

平野 流[1]　米澤 拓郎[1]　廣井 慧[1]　河口 信夫[1,2]

概要：IoT デバイスやオープンデータの普及によりビッグデータが流通している．それに伴う実世界データの入手し易さや，コンピュータの性能向上などにより，大規模かつリアルタイムな人流・交通流シミュレーションの需要が高まっている．大規模なシミュレーションへの試みとして，GPU を利用した高速化が研究されているが，マシン性能に依存した方法であるため，規模の大きさやエージェントの数に限界が生じている．また，処理を分散する分散型シミュレータの研究もなされているが，リアルタイムなシミュレーションが想定されていないことも多く，実世界データを用いたリアルタイムなシミュレーションを行うのは容易ではない．本研究では，複数のコンピュータの使用を想定した，エリアや規模などの構成を柔軟に変更可能な人流・交通流シミュレーションの設計と実装について考案する．具体的には，単一の機能を持ったモジュールをエージェントのドメイン毎やエリア毎に分割し，データ交換プラットフォームを介して連携させることで，分散が容易に可能な設計にした．最後に，これらの仕組みを実装したシミュレータで評価実験を行い，本シミュレータの有用性を検証した．

A Design and Evaluation of Flexible and Scalable Simulator about Person Trip and Traffic Flow

1. はじめに

人や車が多数存在する都市や観光施設などにおいて，人や交通のシミュレーションは重要である．人流や交通流のシミュレーションや分析によって，商業施設や観光施設のサービス向上だけでなく，渋滞の多い地域の混雑緩和，災害時の避難誘導などに応用ができる．また，近年 IoT デバイスの普及やオープンデータの流通，第五世代通信規格 (5G) の実用化などにより，社会のあらゆるところでデータが取得できるようになりつつある．そのような環境の変化をはじめ，計算機器の性能向上などにより，シミュレーションの大規模化，リアルタイム化の需要が高まっている．

しかしながら，現状のシミュレータは単一のコンピュータでの実行を想定したものが多く，エリアやオブジェクト数に限界が生じてしまう．また，大規模化への試みとして分散型シミュレータが開発されているが，リアルタイムなシミュレーションを想定していないものが多く，実世界データを用いたリアルタイムなシミュレーションを行うのは容易ではない．

本論文では，このような課題を解決するために，規模の構成などを柔軟に変更可能かつリアルタイム性のある人流・交通流シミュレータ（以下，本シミュレータと呼ぶ）のアーキテクチャを提案する．エージェントの管理，時刻同期，可視化といった単一の役割をもつ要素をプロバイダと呼び，車や人などのドメインや対象とするエリアで分割し，データ交換プラットフォームを介した連携によって大規模化を実現する．

さらに，実際に複数のコンピュータ上で動作させ，シミュレータの計算処理性能，通信処理性能を評価する．結果として，計算処理速度と通信処理速度がトレードオフの関係であり，特に通信回数の削減が重要であることがわかった．

本論文の構成は次のようになっている．まず，第 2 章で本シミュレータが必要になる理由，課題，研究目的を示す．次に，第 3 章でシミュレーションアーキテクチャの考案とその設計について述べる．そして，第 4 章で現状の実装と本シミュレータの性能実験とその評価をし，第 5 章でまと

[1] 名古屋大学大学院工学研究科
Graduate Scool of Engineering, Nagoya University
[2] 名古屋大学未来社会創造機構
Institutes of Innovation for Future Society,
Nagoya University

めとする．

2. ビッグデータの流通とシミュレーションの新たな需要

2.1 マルチエージェントシミュレーションとその活用

人流や交通流をシミュレーションする手法の一つにマルチエージェントシミュレーション (以下，MAS と呼ぶ) がある．MAS とは，複数の自立したモデルを相互に影響させ，起こる現象をシミュレーションする技術である．特に，人流・交通流は人々の相互のインタラクションによって形成される現象であるため，MAS との相性は良い．そのため，交通流シミュレーション [1][2] や空港における避難シミュレーション [3]，都市部の人流シミュレーション [4] などの研究もなされており，交通や防災，社会科学など様々な場面で活用されている．

社会における人流・交通流のシミュレーションによって様々な事象を解釈できる．例えば，観光施設や商業施設での人の動線や移動パターンなどの分析を行うことで，商品の配置や在庫管理など業務効率化やサービスの質を向上できる．また，空港や祭りなどの混雑しやすい空間において，混雑緩和のために人流を誘導するなどの対策がとれる可能性がある．さらに，災害発生時などの避難シミュレーションなども可能になる．

また，近年 IoT デバイスやオープンデータの普及，5G の実用化に伴い，ビッグデータが流通している．実世界データ入手の容易さや，シミュレーションとして扱う対象の増加，コンピュータ自体の処理性能向上などにより，大規模かつリアルタイムなシミュレーションに対する需要が高まっている．

2.2 関連研究

シミュレーションの大規模化やリアルタイム化の試みとして，様々な研究がされている．例えば，処理能力の高い GPU などを用いて処理能力を向上し，大規模化を実現しようとする研究である．文献 [5] では，エージェントの処理を GPU で行うことで高速化し，大規模化を図った．また，文献 [6] では，GPU による MAS ライブラリの設計と実装を行い，CPU と比べて処理性能の向上を示した．しかしながら，このような GPU を利用したシミュレーションはマシンの性能に依存した設計であるため，オブジェクト数やエリアの規模に限界が生じるという問題がある．

また，一つのマシンを利用するのではなく，複数のサーバに計算処理を分散して連携させながらシミュレーションを行うという研究もなされている．例として，文献 [7] では，High Level Architecture(以下，HLA と呼ぶ) にしたがうシミュレータ基盤の設計を考案した．HLA とは，米国防省の主導で標準化された異種シミュレータが連携するための規格 [8] のことである．HLA では RTI(Run Time Interface) と呼ばれるミドルウェアを通して，分散されたシミュレーションの時刻同期やデータ配信，オブジェクト管理等，6 種類のサービスが規定されている（図 1）．しかしながら，RTI にそれらのサービスが集中しているため，シミュレータの数や規模が大きくなると必然的に RTI にかかる通信量，データ容量が多くなり，オーバーヘッドが発生する可能性がある．また，RTI は共通のサービスと API を備えているものの，ベンダによって通信プロトコルが変わるため，複数の RTI を協調させるのは難しいといった問題もある [9][10]．さらに，既存のシミュレータは，実世界データをリアルタイムに反映するように設計されていないものも多く，その場合リアルタイムなシミュレーションを行うのは容易ではない．

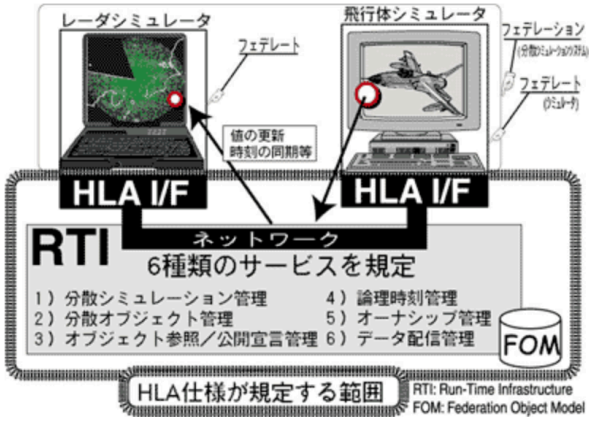

図 1 HLA の概要 [9]
Fig. 1 Consept of HLA

2.3 目的

本論文の目的は，規模や構成などを柔軟に変更可能な人流・交通流シミュレータの設計アーキテクチャを考案すること，またそのシミュレータの性能評価を行うことである．本シミュレータが想定するユースケースは，ビッグデータ活用を想定した都市レベルの超大規模（数十万から数百万）なシミュレーション，商業施設や観光施設でのおもてなし向上を目的とした中規模・大規模（数万から数十万）なシミュレーション，また，災害時の状況に応じた避難誘導などを想定したリアルタイムなシミュレーションである．

2.4 将来的な構想：実世界とサイバー世界の融合

現在，名古屋大学河口研究室ではサービス需給交換基盤「Synerex」[11] を開発している．Synerex とは，社会の需要や供給によって生み出されるサービスから発生するデータを連携させるための基盤である．図 2 のように交通サービスや地域サービスなどで発生するデータの交換によって新しい多様なサービスの提供が予想される．

将来的に，Synerex のようなスマート社会基盤により実

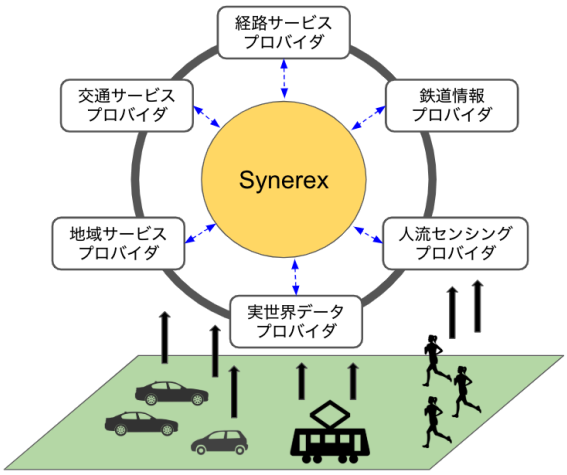

図 2　Synerex の概要
Fig. 2　Consept of Synerex

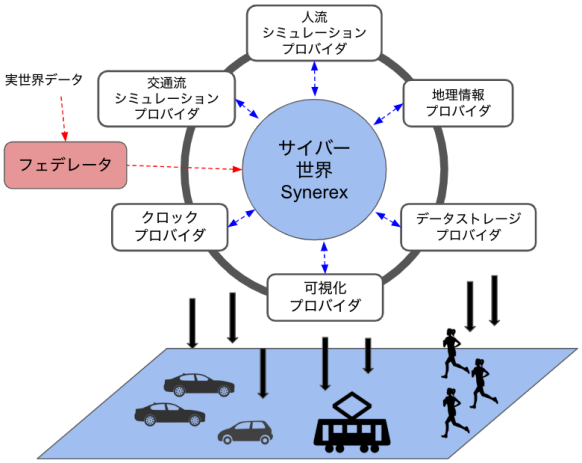

図 3　実世界とサイバー世界の融合
Fig. 3　Fusion of the real world and the cyber world

世界から得られるデータを用いて，コンピュータ内で実世界と同様の世界（サイバー世界）を構築して行う，リアルタイムかつ大規模シミュレーションを構想している．サイバー世界は，図 3 のようにフェデレータを通して得た実世界のデータと，シミュレーションに関わる機能の協調によって構築される．

3.　シミュレータ設計と実装

3.1　プラットフォーム構成

本シミュレータは大規模化，リアルタイム化を実現するため，データ交換基盤を用いたプラットフォームの構成とする．プラットフォームの概要を図 4 に示す．本シミュレータではプラットフォームを構成する単一の役割をもつ要素をプロバイダと呼び，エージェントの管理，データ配信，時刻同期，その他同期処理などを行う．

プロバイダは人や車などのドメインで分けられる．例えば，歩行者プロバイダは歩行者のエージェントを管理し，

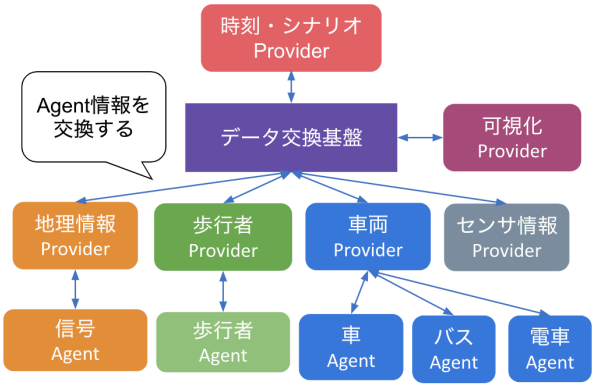

図 4　プラットフォームの概要
Fig. 4　Consept of simulator platform

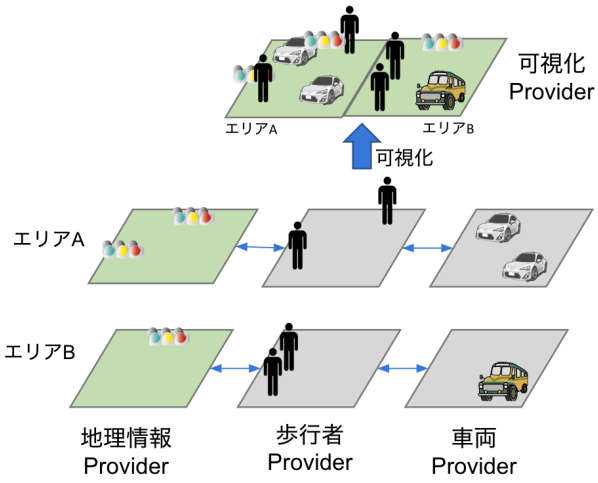

図 5　エリアによるプロバイダの分割
Fig. 5　Dividing providers by area

車両プロバイダは車やバスなどのエージェントを管理する．そして，時刻・シナリオプロバイダからの Step を進める命令によって計算を行う．エージェントを管理しているプロバイダは，計算前にお互いのエージェント情報を共有して計算を行いサイクルを進める．

また，プロバイダは対象とするエリアによる分割が可能である．例えば，図 5 のように，地理情報プロバイダ，歩行者プロバイダ, 車両プロバイダなどの各ドメイン毎に対象とするエリアをエリア A とエリア B で分ける．そして，計算時に隣接するエリアのエージェント情報を共有することによって，矛盾を生じることなくサイクルを進められる．その後，可視化プロバイダによってエリアやドメインの情報を結合し可視化を行う．このようにプロバイダをエリアやドメインによって分割し，それぞれのプロバイダがデータ交換基盤を通して情報を交換しながら，シミュレーションを行っていく．

3.2　データ交換基盤の分散性

分散型シミュレータではいくつかの機能を持ったモジュー

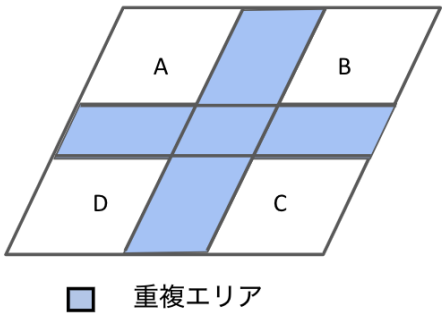

図 6　重複エリアの概要
Fig. 6　Consept of overlap area

ルをデータ交換基盤で連携させてシミュレーションを行う．そのため，モジュールの数が多くなるとデータ交換基盤にかかる通信量やデータ容量にオーバーヘッドが生じてしまう．この問題に対応するためには，データ交換基盤自体の分散性が重要となる．そこで，本シミュレータではデータ交換基盤として Synerex を用いる．Synerex は高度な分散性能を特徴としており，容易に拡張可能であるため，分散型シミュレータに適していると思われる．

3.3　重複エリアによる相互作用の考慮

エリア分割によって生じる問題として，エリア境界における隣接エリアとの相互作用の影響がある [12]．あるエリアの境界にいるエージェントは，隣のエリアにいるエージェントの影響も受けるため，計算を行う前に事前に隣接するエージェントの情報を取得しなければならない．しかし，隣接エリアとの情報の共有にはいくらか通信が発生してしまう．そこで通信回数を削減するために，図 6 のような重複エリアを提案する．各エリア同士を重なるように配置し，その重なった部分を重複エリアと呼ぶ．重複エリア内のエージェントは重複している双方のエリアで管理される．以下に，重複エリアを用いた場合と用いない場合との処理フローと通信量の比較を示す．

[重複エリアを用いない場合]
1. エージェントを初期位置に配置する
2. 隣接エリアと同じエリアの全ドメインの
   エージェント情報を取得する
3. 計算を行う
4. 計算後，隣接エリアの同じドメインの
   エージェント情報を取得し，更新する
5. 2 - 4 を繰り返す

重複エリアを用いない場合は，計算前に隣接エリアとのエリア境界での全ドメインとの相互作用を考慮するため，隣接エリアと同じエリアの全ドメインのエージェント情報を取得する必要がある．そして，計算後にエリア外へ出たエージェントやエリア内に入ってきたエージェントとの整合性を保つため，隣接エリアの同じドメインのエージェント情報を取得し，更新する．よって，一つのプロバイダが 1 サイクルに行う合計通信回数 com は， 隣接エリア数を nei， ドメイン数を dom とすると

$$com = (nei + 1) \cdot dom + nei \cdot 1 \tag{1}$$

と表せる．よって，n 個のプロバイダ全体の総通信量 c は

$$c = \sum_{i=1}^{n} dom_i + \sum_{i=1}^{n} nei_i + \sum_{i=1}^{n} nei_i \cdot dom_i \tag{2}$$

となる．

[重複エリアを用いた場合]
1. エージェントを初期位置に配置する
2. 同じエリアの全ドメインの
   エージェント情報を取得する
3. 計算を行う
4. 計算後，隣接エリアの同じドメインの
   エージェント情報を取得し，更新する
5. 2 - 4 を繰り返す

重複エリアを用いた場合と用いない場合では，2 つ目のプロセスのみが変わっているのがわかる．エージェントを初期位置に置いた時に相互作用に関わるエージェントはすでに保持しているため，計算前に隣接エリアから情報を取得する必要がなくなる．また，エージェントの更新時にも相互作用に関わるエージェントをそのまま保持するため，その後のステップでも同様の計算ができる．そのため，一つのプロバイダが 1 サイクルに行う合計通信回数は

$$com = 1 \cdot dom + nei \cdot 1 \tag{3}$$

となる．よって，n 個のプロバイダ全体の総通信量 c は

$$c = \sum_{i=1}^{n} dom_i + \sum_{i=1}^{n} nei_i \tag{4}$$

となる．以上から，重複エリアの使用によって $\sum_{i=1}^{n} nei_i \cdot dom_i$ 回分の通信回数が削減できる．これは隣接エリアやドメイン数が多くなる程増大していくため，削減効果は非常に大きいと言える．

3.4　エリアによるプロバイダの動的な負荷分散

プロバイダをエリアによって分ける場合，エリアの大きさにはある程度自由度がある．エリアが管理するエージェント数によって計算量が大きく変化するため，エリアの大きさを動的に変える必要がある．エリアを動的に変える方法として，次の二つの方法が考えられる．

一つ目はエリアの大きさを伸縮し，プロバイダの数は変えずに一つのプロバイダが管理するエージェント数を均等にする方法である．この方法は，プロバイダを増やさず，

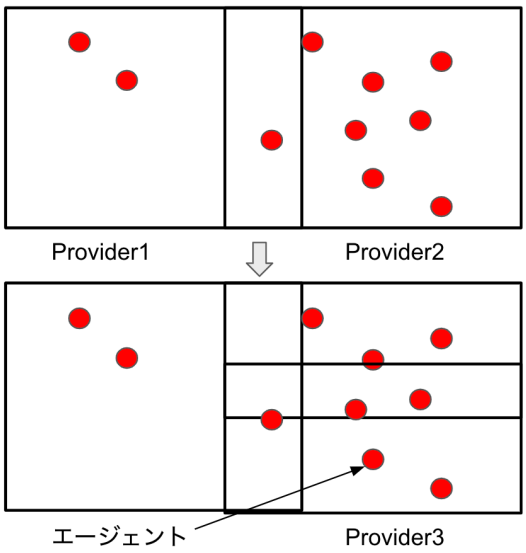

図 7　プロバイダの動的なエリア分割例
Fig. 7　Example of dynamic area division

計算処理を分散できるという利点はあるが，全体的にエージェントが増えた場合に対応ができないといった問題も発生する．

二つ目はエリアを分割する方法 (図 7) である．エージェントが増えたエリアに対して，エリア内のエージェントが均等に分けられるようにプロバイダを増やして分割する方法である．この方法は，プロバイダの増加に従い通信量が増加するが，エージェントが増えた場合でも計算量の分割による処理の分散が際限なく可能となる．通信処理時間と計算処理時間のトレードオフが発生するという問題はあるものの，そのトレードオフを考慮して分割が行えるため，ここではプロバイダの分散によってエリアを分割する方法を採用した．

4.　シミュレータの性能評価

4.1　BSP モデル

本シミュレータを評価する前に，評価指標となり得る BSP モデル [13] について説明する．一般的に，リアルタイムなどの比較的短いステップでサイクルを回す場合，そのステップ時間の間に，エージェントの計算やデータ同期などの処理を終わらせなければならない．そのような課題を効率的に処理するために，シミュレーションの計算・通信・同期の処理をモデル化したものを BSP モデルという．BSP モデルでは，計算処理を「Local Computation (LC)」，通信処理を「Communication (Com)」，同期処理を「Synchronization (Sync)」と呼び，これら三つのフェーズで構成されるステップを SuperStep と定義している．BSP モデルの概要を図 8 に示す．LC にかかる時間と Com にかかる時間が SuperStep を超えると，処理が間に合わず時刻は遅延していく．また，同期処理 Sync がうまく行われなかった場合シミュレーションに矛盾が生じてしまう．このように大規模化，リアルタイム化には，処理時間や通信時間の削減や同期処理の整合性が重要な要素となる．

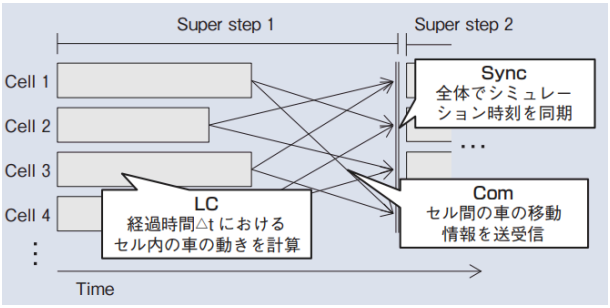

図 8　BSP モデル [14]
Fig. 8　Concept of BSP model[14]

表 1　実験環境
Table 1　Experiment environment

使用した機材等	機材の詳細
通信プロトコル	gRPC(Google Remote Procedure Call)
PC1（サーバ）	MacBook，3.1GHz Dual-Core Intel Core i5
PC2（クライアント）	Vaio，3.30GHz Intel Core i7-6567U
LAN アダプタ	PLANEX GU-1000T
スイッチングハブ	NETGEAR 1000Mbps

4.2　実験方法

4.1 節で説明した BSP モデルのうち，LC と Com にかかる速度評価を行う．実験 1 では一つのプロバイダが管理するエージェント数を増やしていき，1 サイクルでかかる計算処理速度を計測する．エージェントの計算には，Optimal Reciprocal Collision Avoidance[15] という人の回避行動を模したアルゴリズムを実装した．実験 2 では二台のコンピュータを用いて，一方でサーバを起動し，もう一方にプロバイダを複数起動して 1 サイクルにおけるプロバイダ数と通信処理速度の関係を計測する．また，各計測において 10 回ずつ測定し，その平均値と最高値，最低値を記録した．

4.3　実験環境

実験環境の詳細を表 1 に示す．実験 1 では一台のコンピュータ (PC1) を用いた．実験 2 では二台のコンピュータを有線ケーブルとスイッチングハブで接続し，Google の開発した通信プロトコル gRPC を利用し通信を行った．

4.4　評価結果と考察

エージェント数が変化したときの LC の速度測定結果を表 2 と図 9 に示す．エージェント数が多いほど誤差は大きくなるが，ほぼ線形増加という結果が得られた．次に，プロバイダ数が変化したときの通信処理速度の測定結果を表 3 と図 10 に示す．通信処理速度についてはプロバイダ数が

表 2　エージェント数と計算処理速度の関係
Table 2　Relationship between the number of agents and the processing speed

Agent 数	最高値 (s)	最低値 (s)	平均値 (s)
1,000	0.07237	0.04590	0.05700
2,000	0.20347	0.09803	0.11908
3,000	0.25153	0.14329	0.18239
4,000	0.37052	0.22572	0.28159
5,000	0.42631	0.30820	0.33589
6,000	0.52670	0.38389	0.45576
7,000	0.69671	0.49169	0.57708
8,000	0.82980	0.61436	0.71289
9,000	1.07147	0.74554	0.87352
10,000	1.28143	0.91822	1.04942

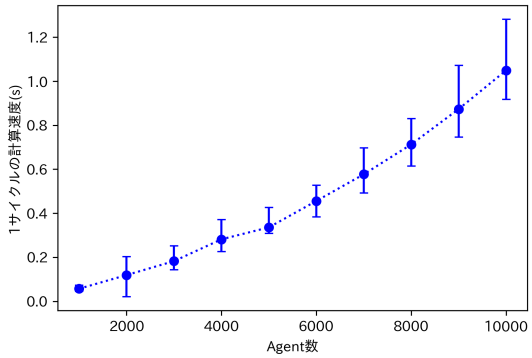

図 9　エージェント数と計算処理速度の関係
Fig. 9　Relationship between the number of agents and the processing speed

表 3　プロバイダ数と通信処理速度の関係
Table 3　Relationship between the number of providers and the communication processing speed

プロバイダ数	最高値 (s)	最低値 (s)	平均値 (s)
1	0.02517	0.011749	0.01761
5	0.02974	0.01296	0.02187
10	0.03877	0.02194	0.02971
15	0.04786	0.02611	0.03453
20	0.07071	0.03054	0.04875
25	0.08324	0.05549	0.07127
30	0.13013	0.06249	0.08733
40	0.25876	0.94210	0.14480
50	0.43028	0.11371	0.28502
60	0.57833	0.22953	0.44682
70	0.94733	0.28450	0.62370

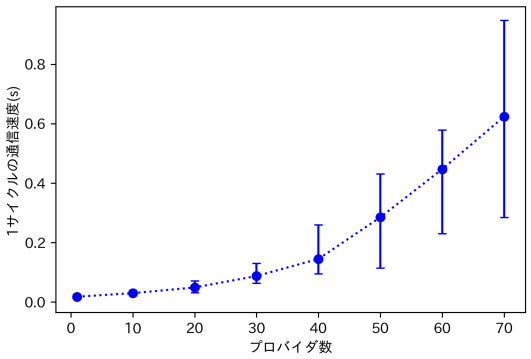

図 10　プロバイダ数と通信処理速度の関係
Fig. 10　Relationship between the number of providers and the communication processing speed

増えるほど，誤差が大きくなり，かかった処理速度の増加率も大きくなっている．

本実験から 1 サイクルにおける計算処理速度と通信処理速度がわかった．計算処理速度はエージェント数に対してほぼ線形であるが，通信処理速度はプロバイダ数に対して二次曲線を描いた．そのため，大規模化の実現には，さらにプロバイダ間の接続数を低減する工夫が必要である．また，エリアを分割し計算量を削減する場合，プロバイダの増加により通信回数が増えるというトレードオフが存在する．本実験の結果は，そのトレードオフから最適な値を見つける良い指標になると思われる．

5.　まとめと今後の課題

本論文では，エリアや規模などの構成を柔軟に変更可能でリアルタイムな人流・交通流シミュレータの設計を考案した．シミュレータの拡張性，分散性を向上し大規模化に対応するため，ドメイン毎，エリア毎にプロバイダを分割し，データ交換基盤を通じて情報を交換し連携をする設計とした．また，エリアの分割によって発生する通信量・計算量の増加を抑えるために重複エリアによる相互作用の考慮と動的なエリア分割を行う仕組みを提案した．さらに，本シミュレータを用いて通信量と計算量の評価を行った．実験の結果，通信量と計算量はトレードオフの関係にあり，計算量よりも通信量の方が速度に対する影響が大きいことがわかった．そのため，今後は通信量の増加を抑制する工夫を考える必要がある．通信量の増加を抑制するには，複数のプロバイダからの通信を一つの通信器が集約し，メッシュによる通信数を削減するなどの対応が考えられる [16] が，まだ検討途中である．また，エージェントの可視化時に可視化プロバイダに情報が集中してしまうという課題も生じる．そのため，可視化プロバイダを分散し，画像として表現した後に一つの可視化プロバイダで差分描画をするという対応が考えられる．今後はこれらの課題に対応していくと同時に，実際の人流・交通流データを用いて，本シミュレータの実用可能性を評価していく必要がある．

謝辞　本研究の一部は，JSPS 科研費 JP17KT0082, JST MIRAI，NICT 委託研究 により支援して頂いております

参考文献

[1] Joachim Wahle and Michael Schreckenberg. A Multi-Agent System for On-Line Simulations based on Real-World Traffic Data. *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, 2001.
[2] Michael Balmer and Kai Nagel et al. Agent-based Simulation of Travel Demand: Structure and Computational Performance of MATSim-T. *2nd TRB Conference on Innovations in Travel Modeling*, 2008.
[3] Jason Tsai et al. ESCAPES - Evacuation Simulation with Children, Authorities, Parents, Emotions, and Social Comparison. *Proc. International Conference on Autonomous Agents and Multiagent Systems*, 2011.
[4] Eric Bonabeau. Agent-based Modeling: Methods and Techniques for Simulating Human Systems. *The National Academy of Sciences*, Vol. 99, , 2002.
[5] Paul Richmond, Simon Coakley, and Daniela Romano. A High Performance Agent Based Modelling Framework on Graphics Card Hardware with CUDA. *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 2, , 2009.
[6] 先山賢一, 芳賀博英. GPU によるマルチエージェント・シミュレーション用 ライブラリ MasCL の設計と実装. 第 7 回社会システム部会研究会, 2014.
[7] Muhammad Usman Awais and Peter Palensky et al. The High Level Architecture RTI as a master to the Functional Mock-up Interface components. *International Conference on Computing, Networking and Communications, Workshops Cyber Physical System*, 2013.
[8] 光行恵司. IT を活用した生産システム開発の効率化・迅速化 : 生産システムシミュレーションを用いたリスクアセスメントと分散型開発のための新たなシミュレーション環境. デンソーテクニカルレビュー, Vol. 9, No. 1, 2004.
[9] 古市昌一, 和泉秀幸. 分散シミュレーションのための統合基盤アーキテクチャ hla の紹介. 情報処理学会, Vol. 41, No. 12, 2003.
[10] 小堀壮彦, 山本一二三. 分散シミュレーション技術の紹介. MSS 技報, Vol. 21, , 2010.
[11] Synergic Exchange. https://github.com/synerex.
[12] 松倉龍之介, 村田嘉利, 鈴木彰真, 佐藤永欣. 自動車のリアルタイム位置管理方法の提案. 情報処理学会第 79 回全国大会, 2017.
[13] Leslie G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 1990.
[14] 小林弘明, 北野雄大, 岡本光浩, 福元健. MAGONIA（分散処理基盤):渋滞予測・信号制御システムへの適用. NTT 技術ジャーナル, 2016.
[15] Jur van den Berg, Stephen J. Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body Collision Avoidance. *14th International Symposium on Robotics Research*, 2009.
[16] Misbah Mubarak, Christopher D Carothers, Robert B Ross, and Philip Carns. Enabling Parallel Simulation of Large-Scale HPC Network Systems. *IEEE Transactions on Parallel and Distributed Computing*, 2015.