並列分散システムにおける大規模交通シミュレーションの性能最適化

金刺 宏樹[1,3]　鈴村 豊太郎[2,3]　松岡 聡[1,3]

概要：大規模交通シミュレーションの需要は増えつつあり，並列分散システムの活用が不可欠となるが，ノードの計算量不均衡による同期の待ち時間発生により，十分な性能向上が得られない問題がある．道路ネットワークに対してグラフ縮退化処理を実行前に施し，また各ノードの計算量に応じて同期頻度を動的に調整する性能最適化手法を適用した結果，ノードに割り当てられるエージェントの個数に不均衡が発生し，高い速度向上率を得ることができなかった．本研究ではこの問題に対処する，エージェントベースシミュレーションの計算量を均等に保つための性能最適化機構として動的ネットワーク分割と同期頻度の削減を提案し，ダブリン市の実データを用いて 23 万台分の車両の移動シミュレーションを TSUBAME 2.5 の 6 ノード 72 コアで実行した結果，最大で 37%の性能向上を実現した．

1. はじめに

近年，防災や都市計画の観点から，大規模な道路交通シミュレーションの需要は増えつつある．また，特に災害時の避難行動などにおいては，今後数時間での交通状況を数分で予測する必要があり，シミュレーション自体の高速化も求められている．このように大規模な道路交通シミュレーションを短時間で行う需要が高まっているが，そのためには 1 台の計算機では実行が困難な場合もあり，並列分散計算機環境が利用されることもある．しかし，交差点や車両などのシミュレーション対象の要素が均等に各計算機ノードに割り振られないと，各ノードで実行時間に不均衡が発生してしまい，同期を行おうとすると膨大な待ち時間が発生することもある．

本研究では，エージェントベースの道路交通アプリケーションを並列分散シミュレーション基盤で実行し，各計算機ノードの不均衡や同期のオーバーヘッドが性能低下の原因であることを示す．その上で，それらの問題に対処するための手法を幾つか提案し，シミュレーション基盤に適用させる．さらに，提案手法を適用させた基盤の性能評価を行うことで，手法の有効性と並列分散シミュレーション基盤で有効な計算機環境の規模について言及する．

[1] 東京工業大学大学院 情報理工学研究科
[2] IBM Research / University College Dublin
[3] JST CREST

2. 大規模道路交通シミュレーション

エージェントベースのアプリケーションを分散環境で実行するために，我々は並列分散プログラミング言語 X10[1], [2] による並列分散エージェントシミュレーション基盤 XAXIS [3] を使用した．XAXIS では定義されているエージェントとして，他のノードへ移動することなく固定される DriverAgent，他のノードへ移動可能で別の DriverAgent に処理されることもある CitizenAgent の 2 種類がある．XAXIS からスレッドを介して実行されるエージェントは DriverAgent であり，CitizenAgent は DriverAgent などを経由して内部状態を変更したり，移動したりする．

また，本研究で使用したアプリケーションの対象として，IBM Research が開発したエージェントベース道路交通シミュレーション Megaffic[4] を XAXIS 上に実装されている．**図 1** に Megaffic へのエージェントの割り当てを示す．Megaffic アプリケーションにおいては，DriverAgent を道路ネットワークの交差点オブジェクト，またこの図では省略されているが，CitizenAgent は個々の車両オブジェクトに対応付けされている．交差点は，流入する道路上にある車両の位置と速度を道路ごとに逐一計算することで，車両の移動を実現させている．

XAXIS では，決められたステップ数だけ DriverAgent の実行，メッセージの送受信，CitizenAgent の移動を繰り返し行う．この時，ノードの間でシミュレーションの整合性を保つ必要があるため，DriverAgent の実行直後と

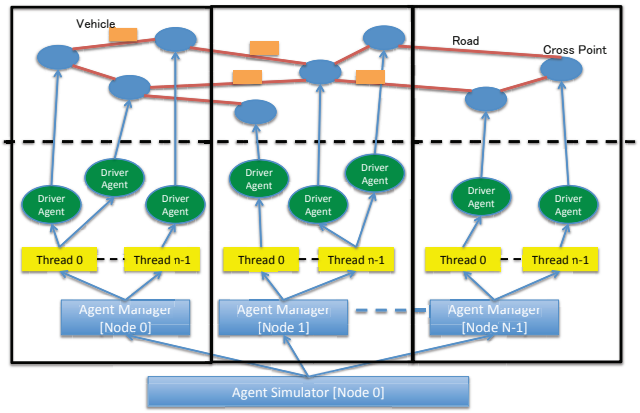

図 1 XAXIS による Megaffic へのエージェント割り当て

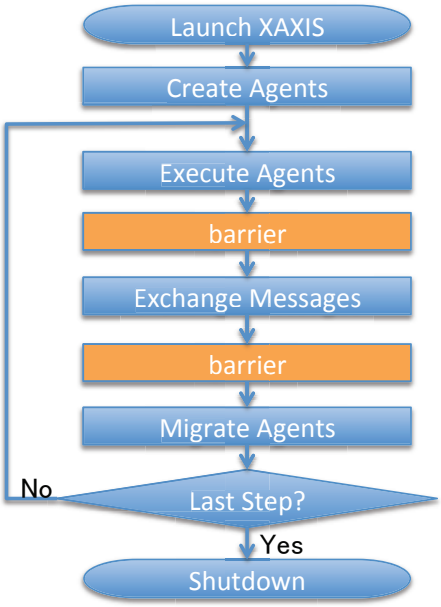

図 2 XAXIS の処理の流れ

CitizenAgent の移動直前に同期を行い，全てのノードの実行を待つようにしている．ここで同期を取ることで，あるノードの車両が他のノードの交差点へ移動する場合でも，シミュレーションステップは同じであるため，時間がずれることなく次の交差点へ移動することが可能である．**図 2** にシミュレーションの流れを示す．

3. 提案手法

3.1 性能上の問題点

使用する計算機のノード数が増加すると，それぞれのノードが扱うエージェント（交差点，車両）の数は減少するため，実行時間が減少すると期待できるが，幾つかの問題点がある．

まず，車両が頻繁に移動することで，他の計算機ノードが処理する交差点に移動することもしばしばある．この場合，車両オブジェクトは一度バイト配列にシリアライズされ，移動先の計算機ノードに送信された後，再び車両オブジェクトにデシリアライズされる．さらに，各ノードが保持する車両について整合性を保たなければいけないため，送信の間は移動元と移動先のノードで同期を取らなくてはいけない．この通り，他ノードへの車両の移動には計算，通信，同期のオーバーヘッドがかかる．

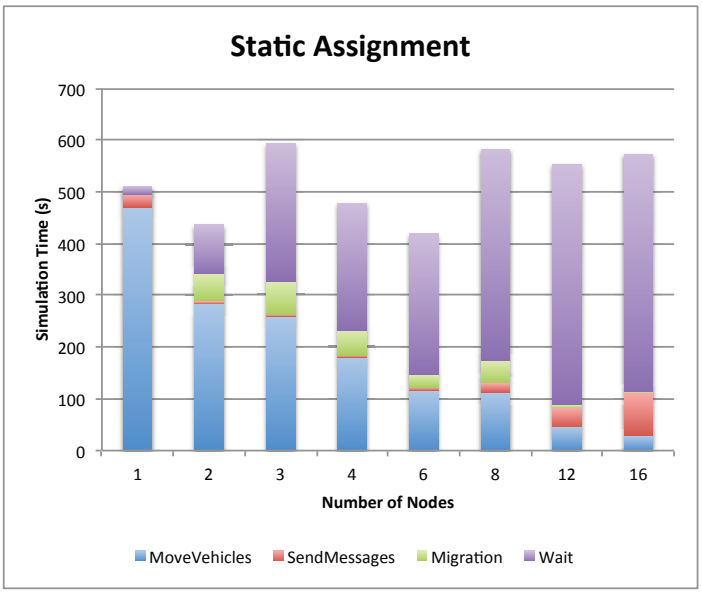

図 3 ノード数変化による実行時間の内訳

さらに，車両が別のノードに移動することで，ある少数のノードに車両が集中する場合もある．シミュレーション開始時に車両台数が均衡していたとしても，シミュレーションのステップが経過するにつれて混雑する場所と空いている場所がはっきりと別れる場合が多い．すると，各ノードに割り当てられている交差点数が同じであったとしても，交差点は車両の移動に計算量のほとんどを費やすため，全体的にノード間で実行時間の不均衡が発生してしまう．

実際に，予備実験として後述する 4.1 の条件で 16 ノードまで実行したところ，最も実行時間が短かったのは 6 ノード使用した場合であり，8 ノード以上ではほとんど変化がなかった．**図 3** では全体の実行時間とその処理の内訳として，上から他の計算機ノードの待ち時間（Wait），車両が他のノードへ移動する際に要した時間（Migration），車両が送信したメッセージの送信時間（SendMessages），DriverAgent に相当する交差点が車両を動かした時間（MoveVehicles）を示している．

交差点が車両を移動させる時間はノード数が増えるに従って減少しており，16 ノードでは 1 ノードのほぼ 16 分の 1 の時間で完了している．しかし，これはあくまでも平均的な実行時間であり，実際は 16 ノードでは全体の実行時間のうち 80%が他の計算機ノードの処理の終了を待つ時間として費やされている．その他，2 ノード以上では車両の他ノードへの移動にかかるオーバーヘッドが見られ，通信のコストも顕著となっていることが分かる．

3.2 交差点の動的割り当て

各計算機ノードに割り当てられる車両（CitizenAgent）の数を均等にするために，それぞれのスレッドやノードに割り当てる交差点（DriverAgent）をシミュレーション途中で変更できる改良を行った．交差点の割り当て先の決定は，事前に同じ条件で実行したシミュレーションのログを解析することにより，なるべく均等になるようにすることで実現した．この場合，同じシミュレーションを 2 回以上実行する必要が出てしまうため，今後の課題でも述べる通り割り当てを自動化させる機構などが必要となる．

3.2.1 ノード内の割り当て

図 1 で示した通り，XAXIS のエージェントに対応する交差点は計算機ノード内のスレッドが分担して処理を行っている．そこで，まずノード内のスレッドに関して車両台数が均等になるよう，スレッドに交差点を選ばせて実行するようにした．従来の XAXIS では車両台数ではなく交差点の数が均等になるようにしていたが，車両の分布に偏りのあるシミュレーションでは不均衡が起こりやすくなるため，より柔軟に均衡が保てるようになると考えられる．

この手法で新たに生じるオーバーヘッドは，各スレッドが担当する交差点のオブジェクトを探し出す処理だけであり，ノードを間で車両を移動させたり交差点の状態を変更する必要は無い．

3.2.2 ノード間の割り当て

前述した方法では，ノード内のスレッド間においては処理量の不均衡が緩和されるが，車両の位置の偏りによるノード間の不均衡は残ったままになる．そこで，交差点においても他の計算機ノードに動的に割り当てられるように処理を変更した．

ただし，交差点を移動させた場合，直接接続する道路オブジェクトも他のノードへ移動させるなどネットワークの構造を変更しなければいけなくなる．そこで，一旦完全な道路ネットワークを各ノードに作成し，スレッドに一部の交差点のみを割り当てることで有効もしくは無効にするようにした．交差点の割り当てを変更する場合は，**図 4** のように，以下の手順で交差点の無効化と有効化，車両の移動を実行する．

(1) 別の計算機ノードに割り当て対象となる交差点にスレッドを割り当てないことで無効にする．
(2) 該当する交差点が保持していた車両オブジェクトを，割り当て先の計算機ノードに移動させる．
(3) 移動先の交差点に新たにスレッドを割り当て有効にし，移動した車両を動かす．

この場合，一つの交差点を別のノードに割り当てる際に，該当する交差点のオブジェクトおよび接続する道路オブジェクトを送信して作成し直す必要はないが，その代わりに全てのノードにおいて完全な道路ネットワークを予め作成しておかなければならず，メモリ使用量に関してオーバーヘッドがある．また，割り当て直す対象となった交差点が車両を保持していた場合は，該当する車両オブジェクトを送信しなくてはならないため，通信のコストはノード内での割り当てと比較して増加すると言える．

しかし，今回の場合はノード間の不均衡のオーバーヘッドが際立って大きく，特に 4 ノード以上では全体の実行時間の半分以上を同期の待ち時間が占めているため，通信量が増えてもこの手法を適用する効果はあると考えられる．

3.3 同期頻度の削減

各ノードで交差点を介した車両の移動およびノード同士で車両を交換する処理時間に不均衡がある場合，処理量が少なく短時間で終了したノードは，図 2 の通り処理量が多いノードの終了を同期によって待たなくてはならず，ノード数増加による高速化の効果が得られにくくなる．

そこで，同期を適用する頻度を 1 ステップごとではなく，10 ステップ，100 ステップに 1 回とすることで，ノードの余分な待ち時間を減らす処理を追加した．XAXIS の同期は X10 の計算機ノードのグループを管理する Team のライブラリのメソッドとして定義されており，このメソッドをシミュレーションステップのカウンタに応じて実行するかどうか判断するようにした．[5]

4. 性能評価

提案した 2 種類の性能最適化手法の有効性を検証するため，Megaffic アプリケーションを載せた XAXIS 基盤と実際の道路ネットワークと乗用車のトリップデータを用いて性能評価を行った．

4.1 実験条件

実験では，TSUBAME 2.5 の Thin-node を最大 16 ノードまで使用した．この Thin-node には, 6 コアの Intel Xeon X5670 CPU が 2 個（ソケット）搭載されていることから，本実験では 1 ノードあたり 12 スレッドを計算用として起動させるようにした．プログラムで使用する Java のヒープメモリは，各ノードで 48GB を確保した．また，計算機ノード間は QDR InfiniBand により，最大 80Gbps の通信が可能である．また、使用したプログラミング言語 X10 のバージョンは現時点で最新である 2.5.1 であり、Java Back-End としてコンパイルした。また、Java のバージョンは 1.7.0 HotSpot である。

使用した道路ネットワークは，Open Street Map (OSM) のアイルランド・ダブリン市とその周辺のデータおよび Dublin Bus の GTFS データを組み合わせることで生成した．ただし，OSM の道路ネットワークは曲線を細かい直線の集合として表現しているため，そのままシミュレーションで使用すると必要なヒープメモリ量が増えてしまい 1 ノードでの実行が困難になる．そこで，分岐がない一本道

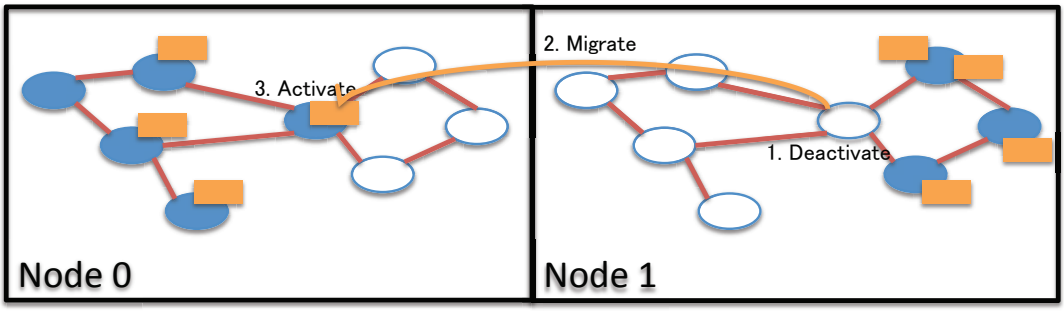

図 4 ノードをまたいで交差点を割り当てる手法

の道路列をまとめて一本の道路とする縮退化を行うことで，効率化を目指した．この場合，曲線道路が直線とみなされるが，道路ネットワークの構造自体は変わらず，まとめた道路の長さも全て合計しているため，シミュレーションへの影響は少ないと言える．縮退化を行ったことで，交差点数は 118,856 から 19,062，道路本数は 269,869 から 40,965 とそれぞれ 6 分の 1 以下に削減することができた．

シミュレーションで走行させた車両のトリップは，ダブリン市のセンサスデータを使用した．このデータは，朝の通勤時間帯に走行した乗用車の始点，終点，出発時刻が 234,378 台分記録されている．この始点と終点の位置から，生成した道路ネットワーク上の最短経路を移動時間を最短化するようにそれぞれ算出し，トリップデータを車両オブジェクトに記録するようにした．

また，実行したシミュレーションのステップ数は 14,400 とした．これは，シミュレーションの 1 ステップが現実社会の 1 秒に対応し，平日の通勤時間帯の午前 6 時から午前 10 時までをシミュレーションの対象とするためである．

4.2 提案手法の効果

シミュレーション実行時に動的に割り当てる手法を適用し，1 ノードから 16 ノードまで実行した結果は**図 5** の通りになった．各ノード数の結果について，左から動的な割り当てを行わない場合（Static），各ノード内でのみエージェントを割り当てる場合（InnerNode），ノードをまたいでエージェントを割り当てる場合（AcrossNodes）を示す．

最もシミュレーション時間が短く済んだ条件は，6 ノードにおいてノードをまたいで割り当てる場合であり，同じノード数においても，各ノードの処理量の不均衡による待ち時間（グラフの Wait）がほぼ半分に減少したことで，動的な割り当てを行わない場合と比較して全体的に 37%の性能向上を実現した．しかしながら，8 ノード以上では他の割り当て条件と同様にシミュレーション時間が延びてしまい，ノード数増加による速度向上率増加の傾向に変化はなかった．特に 16 ノードでは，最も高速な条件である AcrossNodes の場合で比較すると，6 ノードの場合よりも 68%長く時間がかかっていた．

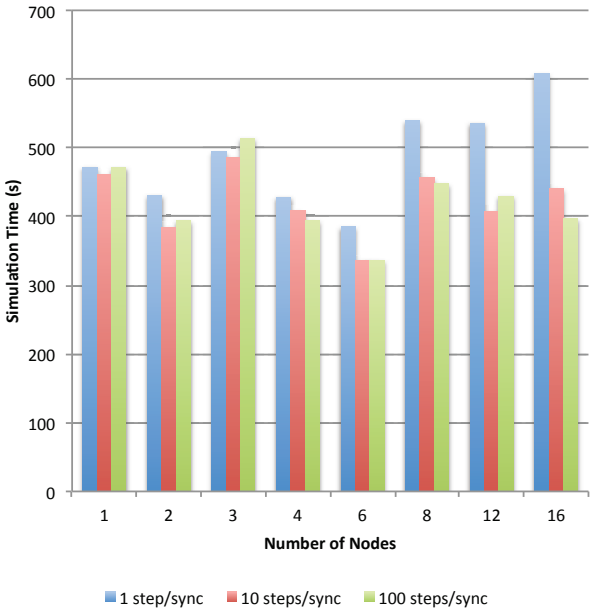

図 6 同期頻度削減による実行時間の比較

また，同期頻度を削減した場合の実行時間は以下の**図 6** の通りになった．

この場合，よりノード数の多い実行で同期頻度削減の効果が得られており，特に 16 ノードでは最大でおよそ 35%の時間短縮となったが，それでも 6 ノードでの実行時間が最も短く，この手法でも速度向上率増加の傾向は変わらなかった．

また，4 ノード以下ではこの手法による実行時間短縮の効果は得られず，3 ノードにおいては 4%ほど実行時間が延びてしまった．図 3 の通り 2 ノード以上ではすでに計算負荷の不均衡が見られており，この手法を適用したことで全体的に不均衡のオーバーヘッドは多少削減できたものの，実行時間の大半がこの類のオーバーヘッドであることを考慮すると，完全には解消できていないと言える．

5. 関連研究

一般に使用されている道路交通シミュレーションのアプリケーションとして有名なものは，SUMO[6], [7] や MATSim[8] などがある．いずれも大規模な道路交通シミュレー

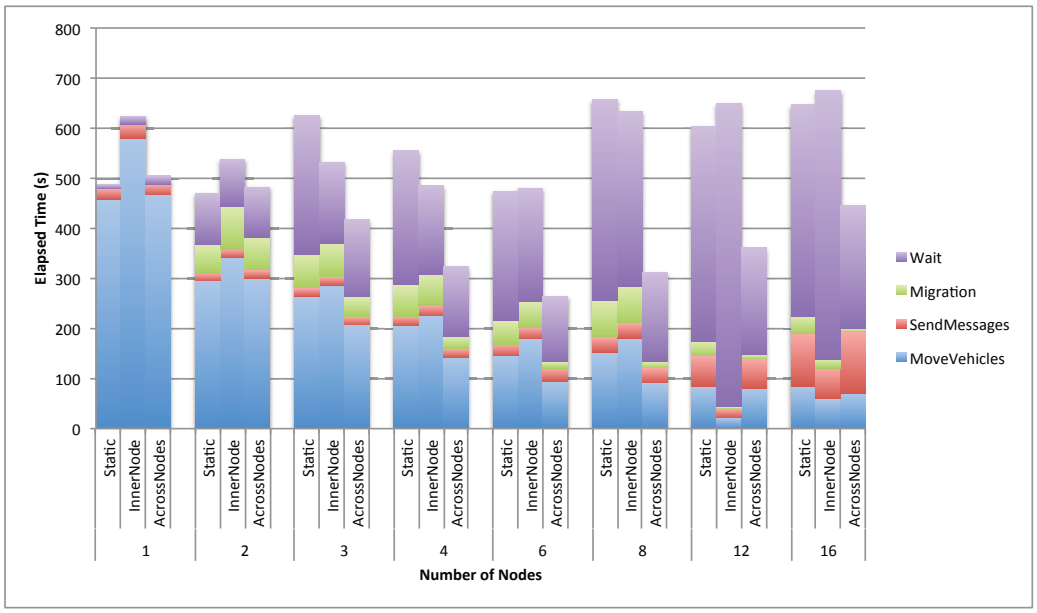

図 5 動的なエージェント割り当てによる実行時間の比較

ションを実行できる特徴を持っており，複数ノードや複数スレッドで性能向上を目指す研究が数多く行われている．

Quentin [9] らが行った研究では，SUMO に通信用インタフェースを実装することで分散環境用に拡張し，ダブリン市の主要な道路ネットワークを使用して性能評価を行っている．しかし，この研究では実験対象の車両台数が 3,000 台程と少なく，また使用した計算機も 2 台にとどまっている．

[10] では，Rashid らが MATSim のメッセージ交換モデルを改良することで，8 スレッドでおよそ 6 倍の速度向上を実現している．ここでは 128GB のメモリが搭載された大規模な 1 台の計算機が使用されており，分散並列環境での実装および性能評価は行われていない．

我々は，先行研究 [11] で実施したような大きな道路ネットワークと車両数での性能評価を行った．

6. 結論と今後の課題

本研究により，我々の提案した性能最適化手法は 6 ノードで最大の性能向上を実現し，ノード間の交差点割り当てと同期頻度の削減の 2 種類の提案手法を適用させたことにより，それぞれ 37%，35%の実行時間短縮を実現した．

今回使用したダブリン市の道路ネットワークと通勤用のトリップデータの規模においては，提案した最適化手法の適用の有無にかかわらず，6 ノード 72 コアが最適な環境であるという結論に至った．これは，問題サイズが固定であり，ストロングスケーリングであるため，マイグレーションなどのコストが高くなり，6 ノードで性能が飽和してしまう．8 ノード以上での実行も試したが，性能は 6 ノードまでで飽和した．

今後の性能面での課題としては，より大規模な道路ネットワークでの性能評価を行い，本研究で提案した性能最適化手法の有効性の検証と最適な計算機ノード数の評価が挙げられる．

また，今回提案した交差点の動的な割り当てに関しても改良の余地がある．現時点での方法では，ノード内でのスレッドの割り当て，ノード間での交差点の割り当てのいずれも，割り当て元の交差点と割り当て先のスレッドまたはノードの場所を全て事前に指定しなければいけない．交差点の割り当てを先を決めるために事前にログを取らなくてはならず，実行時前に静的にログを取得して、グラフ分割をする必要がある．そこで，動的に各ノードの負荷をモニタリングするなどして，自動的に交差点の再割り当てを行えるようにすべきであると考える．

謝辞 本研究は，JST-CREST の研究課題「EBD：次世代の年ヨッタバイト処理に向けたエクストリームビッグデータの基盤技術」の支援による．また，本研究にあたり，性能最適化に関して助言を下さった，松岡研究室の三浦 信一 特任助教に感謝の意を表する．

参考文献

[1] Vijay, A. S., Olivier, T., David, G., David, C., Mikio, T. and Benjamin, H.: A Brief Introduction To X10 (For the High Performance Programmer), The IBM Corporation (online), available from ⟨http://x10.sourceforge.net/documentation/intro/latest/html/⟩ (accessed 2015-02-05).

[2] Vijay, S., Bard, B., Igor, P., Olivier, T. and

David, G.: X10 Language Specification Version 2.5, The IBM Corporation (online), available from ⟨http://x10.sourceforge.net/documentation/languagespec/x10-latest.pdf⟩ (accessed 2015-02-04).

[3] Suzumura, T. and Kanezashi, H.: Highly Scalable X10-Based Agent Simulation Platform and Its Application to Large-Scale Traffic Simulation, *Distributed Simulation and Real Time Applications (DS-RT), 2012 IEEE/ACM 16th International Symposium on*, pp. 243–250 (online), DOI: 10.1109/DS-RT.2012.44 (2012).

[4] Takayuki, O., Takashi, I., Hideyuki, M., Toyotaro, S. and Tsuyoshi, I.: Toward simulating entire cities with behavioral models of traffic, *IBM Journal of Research and Development*, Vol. 57, No. 5 (2013).

[5] Suzumura, T. and Kanezashi, H.: Accelerating Large-Scale Distributed Traffic Simulation with Adaptive Synchronization Method, *20th ITS World Congress 2013* (2013).

[6] Krajzewicz, D., Hertkorn, G., Rssel, C. and Peter, W.: SUMO (Simulation of Urban MObility) - an open-source traffic simulation, *Proceedings of the 4th Middle East Symposium on Simulation and Modelling (MESM20002)* (2002).

[7] Hilbrich, R.: DLR - Institute of Transportation Systems - SUMO - Simulation of Urban MObility, Institute of Transportation Systems, Berlin (online), available from ⟨http://www.dlr.de/ts/en/desktopdefault.aspx/tabid-9883/16931_read-41000/⟩ (accessed 2015-02-04).

[8] Nagel, K.: Agent-Based Transport Simulations — MATSIM, matsim.org (online), available from ⟨http://www.matsim.org⟩ (accessed 2015-02-05).

[9] Quentin Bragard, A. V. and Murphy, L.: dSUMO: Towards a Distributed SUMO, *The first SUMO User Conference (SUMO2013)* (2013).

[10] Rashid A. Waraich, David Charypar, M. B. and Axhausen, K. W.: Performance Improvements for Large Scale Traffic Simulation in MATSim, *9th Swiss Transport Research Conference* (2009).

[11] Toyotaro, S., Charuwat, H. and Hiroki, K.: Towards Billion-Scale Social Simulations, *Winter Simulation Conference 2014* (2014).