特別研究報告書

社会的インタラクションが交通に与える影響

指導教員　　石田　亨　教授

京都大学工学部情報学科

冨田　典也

平成22年2月3日

社会的インタラクションが交通に与える影響

冨田　典也

内容梗概

人々は，日常的な社会活動において，家から職場まで通勤する，家から学校まで通学する，店まで買い物に行くなどの移動を行う．一般的な人は，移動をするときは，より早く目的地に着けるような経路を選択することを好む．そのために利用する情報には，経路の長さなどの静的な情報だけでなく，情報機関や他人から伝えられる動的な混雑状況もある．

外部から与えられる交通情報には，全ての車両の移動状況を基にした不特定多数の人が入手できる道路交通情報の他に，自分と同じ性質を持つ他人とのコミュニケーションを通じて入手できる，特定のドライバーが道路を通過したときの混雑状況の情報もある．本論文では，このような情報交換を社会的インタラクションとして扱う．

これまでの交通シミュレーションでは，全ての車両が共通の道路交通情報を入手できるという状況を考慮したものが多く，そのことが引き起こす問題点と改善策に対する研究も数多く存在する．しかし，各車両が車両間の属性によって限定された情報を利用するという要素を入れたものは少なかった．

そこで，本論文では，社会的なインタラクションが交通に与える影響を調べるためのシミュレーション環境の設計と実装を行う．そのために，各車両が過去の移動履歴を持ち，社会的なインタラクションをとることができる車両にだけ，その移動履歴を伝達するという状況を設定した．

本論文では，広域的な交通状況をシミュレートできる，MATSim という交通シミュレータに社会的インタラクションの機能を実装する．MATSim では，各車両が 1 日ごとに行動計画を定め，その計画に従って移動と滞在を繰り返し，移動にかかった時間を基に計画の評価を行う．行動計画は，自身の持つ計画から，その評価値に従って決定されることが多い．しかし，他の車両の移動時間を使って最適と思われる目的地までの経路を探索し，新たな行動計画を作ることもある．このとき，もともとの MATSim では，全ての車両の通過時間の平均を使用する．ここで本論文では，各車両が，車両ごとに異なる複数の別の車両を情報源として持っていて，これらの情報源の通過時間がある場合は，それを優先して使用するように変更する．これらの情報源が，社会的なインタラクションに

よって情報交換ができる車両に相当する．

社会的インタラクションの効果が正しく導入されたことを示すために，自身の効率を優先する車両が他者に悪影響を及ぼすことを，シミュレーションによって確かめた．具体的には，自分が利用する道路を他の車両が利用しなくなるように虚偽の情報を流す車両がいる場合，虚偽の情報配信をする車両と通常の情報を流す車両の移動時間が，それらの割合に応じてどのように変化するのか調べた．その結果，虚偽の情報を流す車両の割合が増えることで，虚偽の情報を流す車両と通常の情報を流す車両の両方の移動効率が悪化した．つまり，今回の設定では，虚偽の情報を申告することの利点はないと言える．

Effect of Social Interaction on Vehicular Traffic

Fumiya TOMITA

Abstract

People move in social activities. For examples, they go from home to working places, to schools, or to shops. When common people move, they prefer to select a route to get to their destination faster. The information used in order to get to the destination fast is not only static one such as the length of the route, but also the dynamic one from agencies and others.

In traffic information given from the outside, in addition to road traffic information based on the movement of all vehicles, which is available on an unspecified number of people, there is road congestion when other driver passed this road, which can be obtained through communication with others. In this paper, exchanging information like this is regarded as social interaction.

Many of existing traffic simulations considers a situation where all vehicles can obtain common road traffic information. Therefore, there are many researches about the issues of using common information and remedy for that. However, there are few ones which consider that each vehicle uses information that is limited by the attributes between vehicles.

In this study, I designed and implemented a simulation environment to examine the impact of social interaction on vehicular traffic. To do so, I considered the situation where each vehicle had history of moving and gave it to others which took social interaction to the vehicle.

In this study, I implemented the features of social interactions in MATSim. This is a traffic simulator that can simulate the broad-based traffic. In MATSim, each vehicle determines a plan of the action each day, repeats the moving and staying in accordance with the plan, and evaluates the plan based on the time it took to move. The used plan is often determined according to the value of its own evaluated plan. However, each vehicle sometimes explores the best route by passing times of other vehicles, and creates a new plan. In this case, in original MATSim, the average of passing time of all vehicles is used. In this paper, each vehicle has other vehicles as sources of information and prefers to use average of passing time of those sources to that of all vehicles. These sources

are equivalent to the vehicles to which each vehicle can take social interaction.

To demonstrate that this simulator shows the effect of social interaction properly, I simulated that the vehicle which gives priority to its efficiency affects others. Specifically, I examined that if there were some vehicles that gave false information to others so that other vehicles didn ’t use the roads they will use, how transit time of the vehicles which gave false information and the vehicles which gave usual information changed according to those percentages. In this experiment, when the percentage of the vehicles which gave false information was increased, transfer efficiency of both of the vehicles which gave usual information and which gave false information was aggravated. In other words, it can be said that there are no advantages of giving false information about this setting.

社会的インタラクションが交通に与える影響

目次

第1章　はじめに

広域的な交通シミュレーションは，新たな施設ができる，工事などで道路が封鎖されるなどの世の中の変化によって，交通状況がどう変化するかを予測するために，よく利用されている．そのため，交通シミュレーションには，現在あるいは未来の交通状況を正確に再現することが求められる．再現性の向上のためには，交通状況に影響を与える要因を入れることが重要となる．

交通シミュレーションにおける一般車両は，速く目的地にたどり着けるように，学習と経路選択を行うことを前提としているものが多い．学習のためには，自身の経験や外部からの情報がよく利用される．そこで，シミュレーションにおける外部からの情報の収集方法と，経路選択時の利用方法について考える．

本論文では，仕事などで何度も同じ目的地に行く状況をシミュレーションで再現する．日常的に通っている場所に行くときは意図的に交通情報を収集することが無く，情報が得られるとしても，普段利用している道路の方が迷いにくいため，ほとんどの場合，自分が通ったことのある経路を利用すると思われる．しかし，他の経路の方が速いと考えて，外部からの情報を基に，新たな経路を選択することもある．その時の経路選択の指標には，一般的に公表されている道路交通情報を利用するのが確実であり，利用人数も多いと思われる．しかし，中には特定の人との社会的なインタラクションにより，特定の車両が通った時の道路の混雑状況を収集し，それを道路交通情報よりも優先して使用する人もいるだろう．

これまでの交通シミュレーションでは，一般的な道路交通情報を想定した，特定の道路における全ての車両の通過時間の平均あるいは通過車両台数を基に学習と経路選択をすることを想定しているものが多く，そのことに対する問題点や改善策についても数多く研究されている．しかし，自分と同じ属性を持った車両との情報交換によって学習を行うという設定は少なかった．

そこで，本論文では，社会的なインタラクションが交通に与える影響を調べるためのシミュレーション環境の設計と実装を行う． そして，このような社会的インタラクションの要素を取り入れることで，交通シミュレーションの内容に影響が出ることを示す．

シミュレーションの実装方法としては，各車両がリンクを通過した時の通過時間を行動履歴として集計するように変更する．そして，各車両は他の車両の

各リンクの通過時間を収集し，車両ごとに指定された他の車両（情報源）の通過時間と，全車両の平均通過時間とを併用して経路選択をするという手法を考える．具体的には，新たな経路を選択するときに，情報源が通過したリンクは，情報源の通過時間を使用し，情報源が通過しなかったリンクは，全車両の平均の通過時間を使用して，目的地まで最も少ない時間で行ける経路を選択する．

この設定において，社会的インタラクションが交通に影響を与える例として，一部の車両が自分の効率を優先する場合のシミュレーションを行う．具体的には，他の車両が自分の通る経路を利用しないように，実際の通過時間より長い通過時間を申告する車両が旅行効率に与える影響を調べる．

第2章　交通シミュレータの概要

本論文で使用するシミュレータは，ベルリン工科大学で開発された大規模交通シミュレータである MATSim の一部を変更し，道路上での詳細なシミュレーションができるようにしたものである．MATSim はエージェントベースの交通シミュレータであるため，エージェントごとに固有の設定をすることが容易であるという点で，今回の手法の実装に向いている．また，過去に MATSim を利用してスイス，チューリッヒ市内とその周辺の交通状況を再現した実績がある [1]．以下にその概要を記す．

2.1　全体の動作

MATSim は，各エージェントの 1 日の行動と学習をシミュレートすることで，交通状況の変化を再現する，マルチエージェント交通シミュレータである．MATSim では，エージェントの移動手段は車だけであり，以下では車両とエージェントは同じものとして扱う．

MATSim では，各車両が 1 日ごとの行動計画をプランとして保持している．プランには，出発地点となるリンクの id とそこを出発する時刻，いくつかの目的地のリンクの id とそこに到着する予定時刻と滞在する時間，及び目的地間の移動時に利用するノードの id などが記録されている．ここでは，プランに含まれるデータの中で，出発地点と出発時刻，目的地と滞在時間を OD データとする．

MATSim では，1 日ごとのシミュレーションを繰り返し，その結果を元に各エージェントが学習を行う．その基本的な動作は図 1 のとおりである．まず，プ

ランに基づいて交通シミュレーションを行い，次に，プランでの到着予定時刻と，シミュレーションでの到着時刻の差などを基に使用したプランに評価値を付け，その後，プランの評価値あるいは行動履歴に基づいてプランを再選択する．以上の操作を指定した循環回数だけくり返す．交通シミュレーションとプランの選択の概要は後述する．

MATSim では，入力ファイルは xml 形式で記述され，データベースに格納されるときに，適切なデータ形式に変換される．図 1 の network.xml には，後述のネットワークに関する情報が記されており，plans.xml には，エージェントが持つプランや走行戦術などの，全てのエージェントに関する情報が記されてい

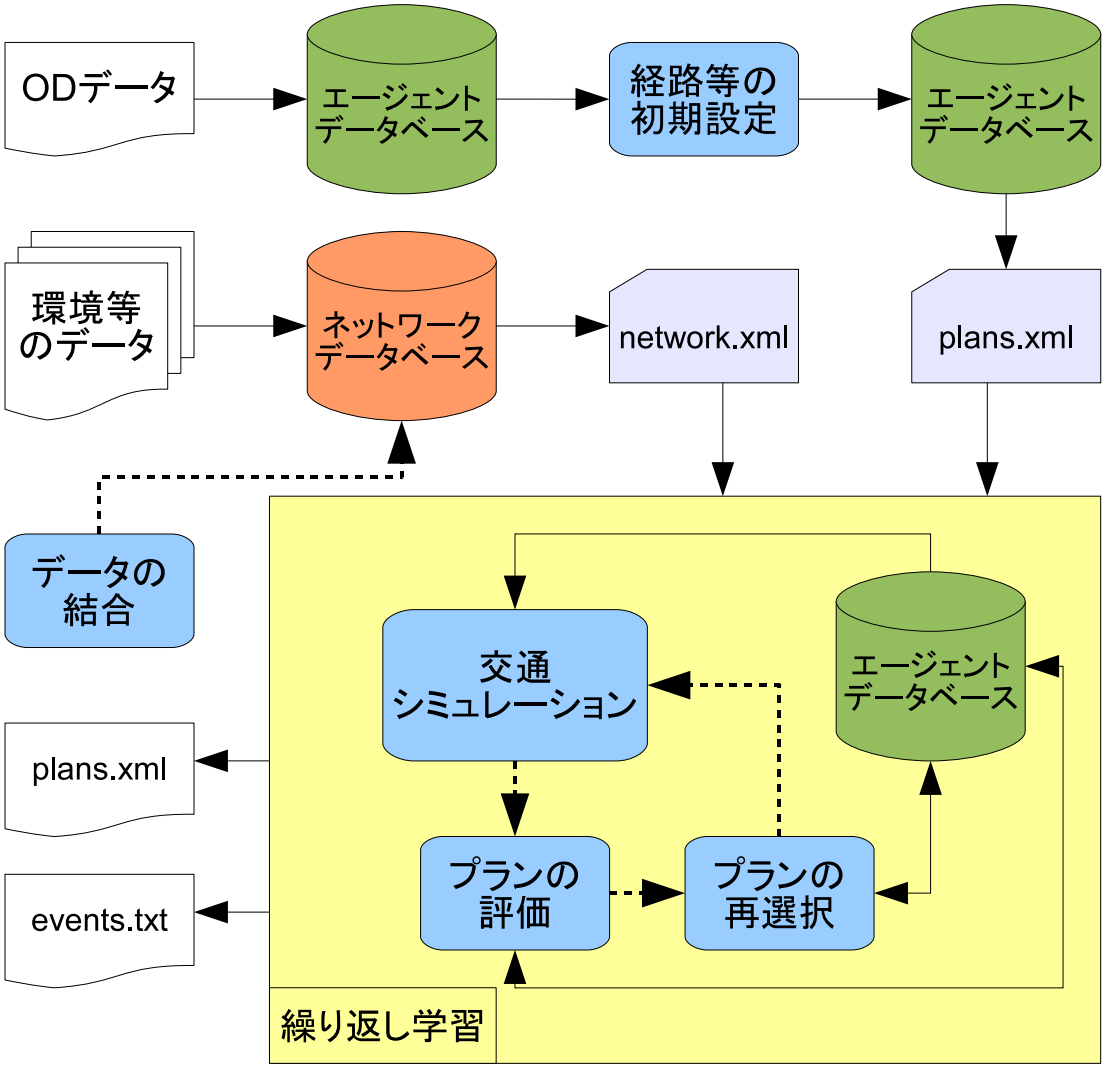

図 1: MATSim の動作概要

る．これらのファイルは，繰り返し学習をするときに，別のデータベースに格納されて，そのデータベースに対して読み書きが行われる．そして，1 回の繰り返し学習が終わるごとに，プランを変更した後の plans.xml ファイルと，後述するイベントについて記載した events.txt を出力する．

2.2 交通シミュレーション

交通シミュレーションで用いるネットワークは，道路を表すリンクと，道路の端点を表すノードからなる．各ノードは，以下の属性を保持している．

- id: ノードの ID.
- x: ネットワーク上での x 座標 (単位:メートル).
- y: ネットワーク上での y 座標 (単位:メートル).

MATSim における交通シミュレーションはリンク上の移動の繰り返しであり，交差点などの通過時間は考慮されないため，ノードの属性は少なくなっている．リンクは全て一方通行である．リンクの持つ属性は以下の通りである．

- id: リンクの ID.
- from: 始点となるノードの ID．リンク間の接続を表すのに用いられる．
- to: 終点となるノードの ID．リンク間の接続を表すのに用いられる．
- length: 進行方向の長さ (単位:メートル).
- width: 進行方向に垂直な方向の長さ (単位:メートル).
- permlanes: 車線数．各車両は，走行したい車線番号を指定することができ，その車線番号の最大値を表す.
- freespeed: 自由速度 (単位:メートル毎秒)．後述の予想通過時間を算出するときに用いられる．
- capacity: 最大交通容量 (単位:台数毎秒)．単位時間あたりにリンクから出ることができる車両の台数を表す.

交通シミュレーションにおいて，各車両は滞在と移動をくり返す．滞在状態の車両は，出発地点では指定した時刻になると，それ以外の地点では指定した滞在時間が過ぎると，移動状態に移行する．移動状態の車両は，リンク上での移動をシミュレートし，それが終わると次のリンクに移動する．この動作を繰り返して，目的地となるリンクに着くと，滞在状態となる．

前述のプランでは，滞在状態を act, 移動状態を leg として，それぞれの情報を記憶している．act と leg は，それらを適用する順番に従って，交互に記録さ

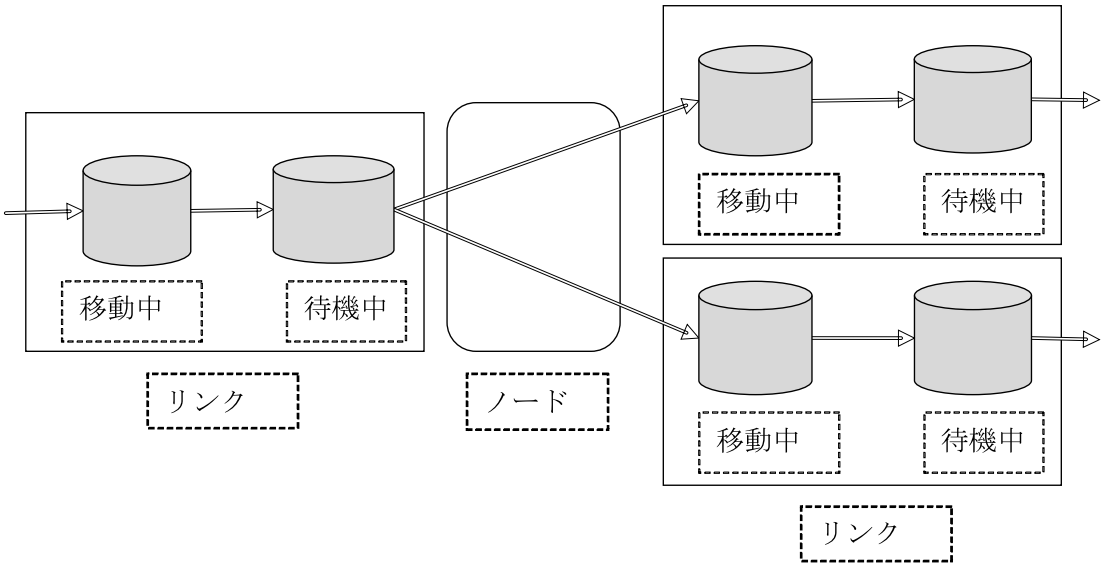

図 2: 車両の移動の様子

れている．act の属性は以下の通りである．

- type: 仕事，学校，買い物，自宅などの滞在の目的．必ず設定される．
- x: 滞在場所の x 座標.
- y: 滞在時間の y 座標.
- link: 滞在場所となるリンクの ID.
- starttime: 滞在中の活動の開始時刻.
- endtime: 滞在終了時刻．最初の act には必ず設定される．
- dur: 滞在中の活動を行う時間．最初以外の act には必ず設定される．

このとき，x と y の組と link のどちらか一方は，必ず設定されなければならない．leg の属性は以下の通りである．

- mode: 移動手段．必ず設定されるが，現在の設定では車のみ.
- deptime: 予想出発時刻．評価値を付ける時に用いる．
- travtime: 予想移動時間．評価値を付ける時に用いる．
- arrtime: 予想到着時刻．評価値を付ける時に用いる．
- route: 移動のときに通過するノードの ID の列.

図 2 に車両の移動の概要を示す．移動中の車両は，車両ごとに設定されたエージェントの運転戦略及び車両の性質に基づいて，リンク上での挙動をシミュレートする．それが終了した車両は，待機中のキューに格納される．通常，待機中

の車両はすぐに次のリンクに移動する．ただし，各リンクでは，単位時間当たりのリンクを出ることができる車両数を最大交通容量として指定しており，それを超える車両がいる時は，先に待機状態になっていた車両が次のリンクに移動するまで待つことになる．

リンク上での移動は，従来の MATSim に変更を加えて，微視的なシミュレーションができるようにしてある．具体的には，リンクに入った車両は，リンクの情報を基にした図 3 のような座標上を移動する．リンク内での車両は，x 座標，y 座標と，x 軸方向及び y 軸方向の速度と加速度を保持している．リンクに入ったときの車両の y 座標は 0 で，リンクに入ったときの x 座標は以下のとおりである．

- 出発した直後に入ったリンクの場合，最も左の車線の中央の x 座標.
- リンクからリンクに移動するときは，出る側のリンクでの終端の x 座標値を，出る側のリンク，入る側のリンクの道路幅に比例させた値.

シミュレーション中の各車両は，一定時間ごとに，以下の操作を行う．

- 現在の速度を基に車両の位置を更新する．
- 現在の加速度に従って車両の速度を更新する．
- 現在のリンク内の状況と，車両ごとの運転戦略を基に，車両の加速度を更新する．

車両の y 座標がリンクの長さを超えた時点で，道路上でのシミュレーションは終了となる．なお，現実においてカーブしている道路は，図 3 のリンクをいくつか使用し，少しずつ移動方向を変化させることで表現する．

交通シミュレーション中は，車両の移動に関して，イベントが発生するごとに発生させた時刻とリンクの ID とエージェントの ID とイベントの種類を図 1 の events.txt ファイルに出力する．イベントとは，誰がいつどこで何をしたかを記録したものであり，イベントの種類，車両の ID，リンクの ID．イベントの起こった時刻がセットで記録される．これらの情報を分析することで，エージェントの出発地点や目的地ごとに，旅行時間や特定のリンクの通過台数などを調べることができる．イベントの種類は以下の通りである．

- departure: act を終了し，移動状態に入った．
- arrival: 目的地となるリンクに到着した．
- entered link: 道路リンクに入った．
- left link: 道路リンクを出た．

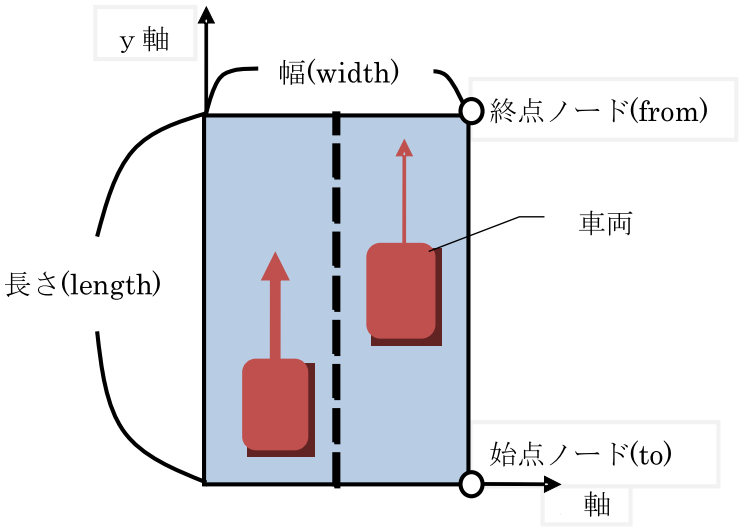

図 3: リンクの概要

- wait2link: 次に移動するリンクの容量が足りないなどの理由で待機していたが，それが解消された．
- stuckAndAbort: 指定したシミュレーション時間を超えたため，シミュレーションを終了した．

また，交通シミュレーション中は，一定時間ごとに各エージェントの位置と速度と進行方向をスナップショットとして出力する．これをビューアで再生することで，交通シミュレーションの内容を目視で確認することができる．

2.3 経路決定戦略

MATSim では，２種類のプランの選択方法が，全ての車両について決まった確率で適用される．一つ目は，自身が持つプランの中で最も評価値の高いものを選択するという方法である．これは，自分が通ったことのある経路から，自分が通った時の情報を基に選択するという戦略に対応する．

もう一つは，最も速く到達できると思われる経路を探索し，その経路を利用するプランを新たに構築する方法である．これが，外部からの情報を基に経路選択をする戦略に対応している．経路探索には，前回の交通シミュレーションでの行動履歴による各リンクの予想通過時間を重みとしたダイクストラ法を使用する．次章で，外部からの情報を利用した予想通過時間の決定方法について述べる．

第3章　社会的インタラクションの導入

本論文では，社会的なインタラクションの形態として，他の車両から経路の通過時間を取得し，自身も通過時間を他の車両に渡す状況を考える．以下にその手法を考える必要性と，シミュレーションにおける設計，実装方法について述べる．

3.1　社会的インタラクションによる情報の共有

近年は，テレビやパソコンなど，外部から情報を受け取るツールが普及したことにより，道路の混雑状況などの情報を容易に入手できるようになっている．このような，不特定多数に向けて配信される交通情報は，多くの人にとって信頼性が高く感じられるため，利用者が多く，交通状況に与える影響も大きい．しかし，人によっては一般的に配信される交通情報を取得できる環境が無い場合があり，道路によっては交通状況の計測をしていないこともあり得る．そのため，このような情報だけを考えるだけでは不十分である．

交通状況に影響を与える情報の入手先は，一般的に配信されている情報だけとは限らない．社会活動におけるインタラクションによって情報を入手し，その情報に従って行動することが，交通状況に影響を与えることもある．社会的なインタラクションの形態は様々だが，本論文では，各人が自分と同じ性質を持つ人とコミュニケーションによって情報を交換することを社会的インタラクションとする．コミュニケーションのためのツールも，近年では広く普及しているため，このような情報は，場合によっては一般的に配信される交通情報よりも入手しやすいと考えられる．

ここで，上で述べた社会的インタラクションを再現するシミュレーション環境を設計する．以下では，シミュレーションにおける各エージェントが社会的インタラクションによって情報を入手することができるエージェントを情報源として定義する．情報交換をする相手は人によって決まっているとすると，人づてに情報が伝わることを考えても，情報を入手できる範囲には限界があると考えられる．そこで，エージェントごとに異なる一定数の情報源を，各エージェントが保持しており，これらの情報源からの情報を，一般的な交通情報より優先して使用するような状況を，交通シミュレーションにおいて再現する．

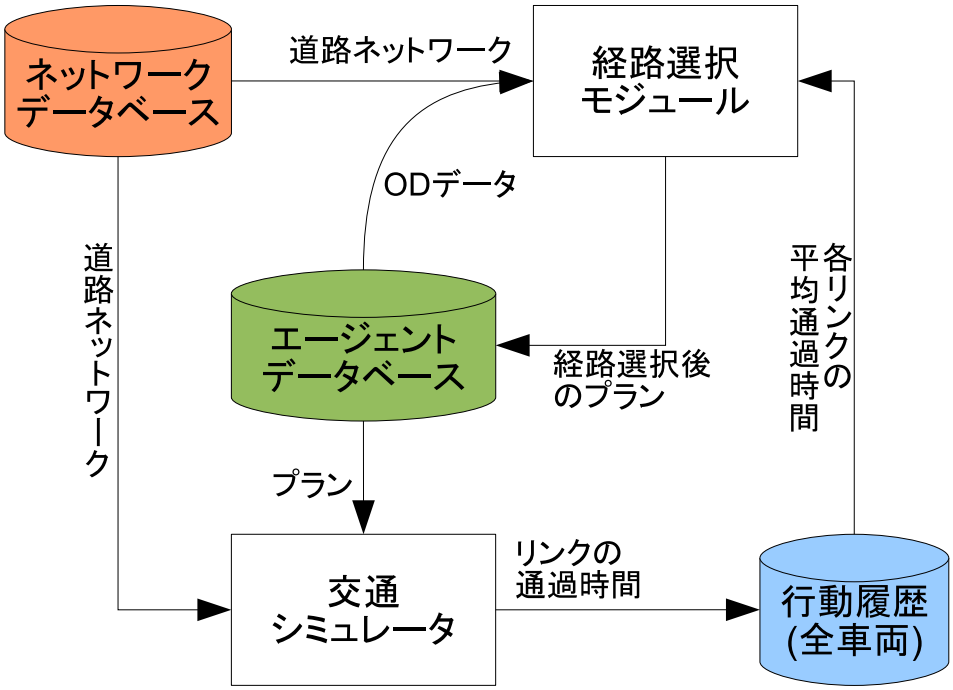

図 4: 変更前の経路選択の模式図

3.2 交通シミュレータによる実装方法

この論文で使用する MATSim における経路選択のもともとの実装は図 4 のとおりである．ここでは，直前のシミュレーション中に車両がリンクを通過するのにかかった時間の合計と通過した車両の台数を，リンクと時間帯ごとに集計している．リンクとは道路ネットワークにおける細分化された道路であり，時間帯は 15 分刻みで指定されている．そして，経路探索をするときに，通過時間の合計と通過車両数から算出したリンクの通過時間の平均をそのリンクの予想通過時間とする．どの時間帯の平均通過時間を利用するかは，目的地の出発予定時間と，これまでに通過するリンクの予想通過時間によって決定する．通過が予想される時間帯にまったく車両が通らなかったリンクは，リンクごとに決められた自由速度で通過できたときの通過時間を、そのリンクの予想通過時間とする．なお，合計通過時間と通過車両数は，全ての車両のプランの再選択後に 0 にリセットされる．

今回の実装における変更内容は，図 5 の通りである．新たに追加した要素として，各車両は情報源となる車両の ID を記憶している．情報源とは，各車両が個別の通過時間を取得することができる車両のこととする．これらの情報源は，社会的インタラクションをとることで情報交換ができる車両に相当する．さら

に，自分が情報源から受け取る通過時間や，他の車両に提供する通過時間に対する補正率を設定することもできる．

各車両は直前のシミュレーションにおいて，リンクを出る時に，従来と同じく合計の通過時間を集計するとともに，自身の通過時間と通過した時間帯を別

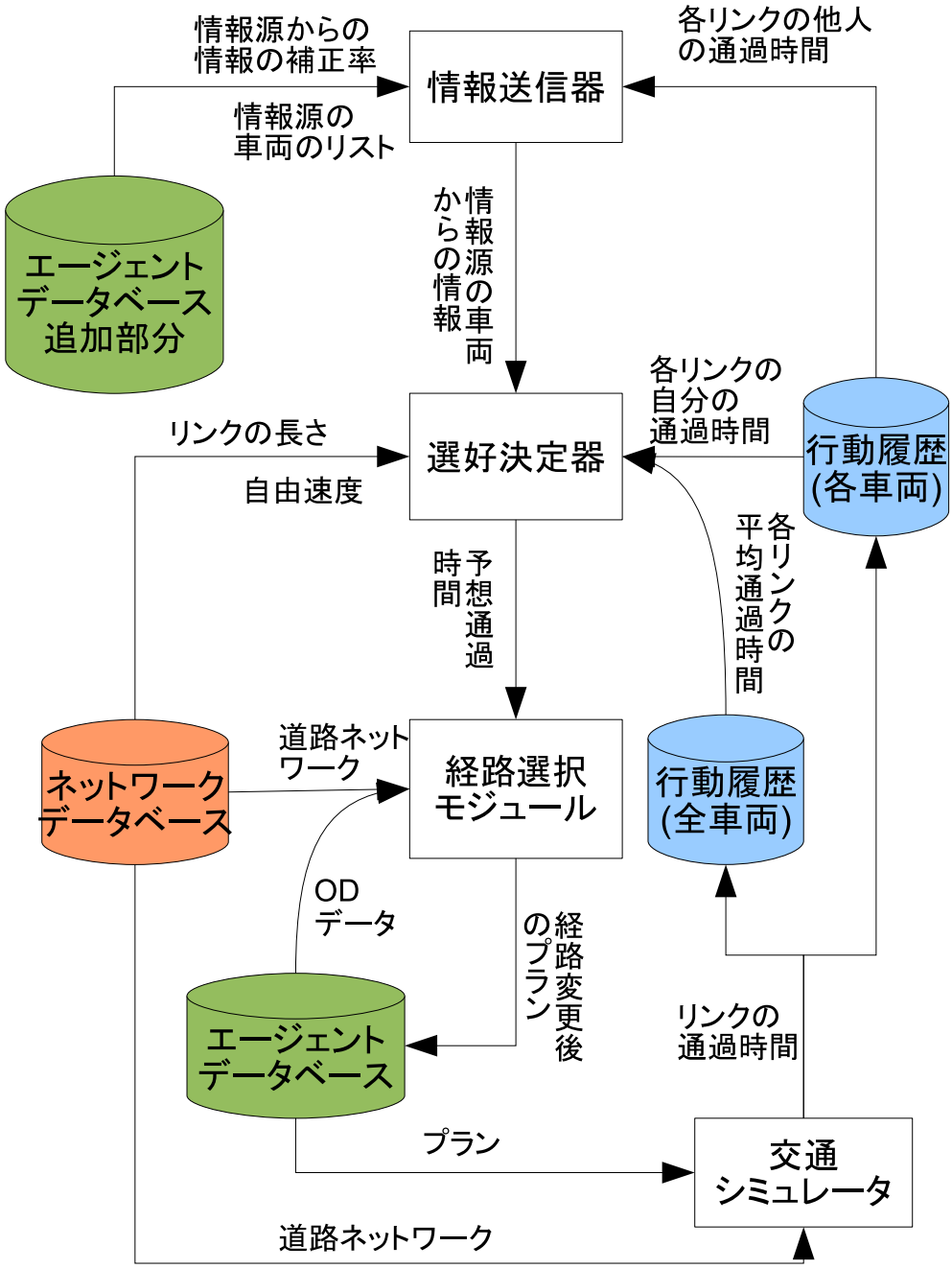

図 5: 社会的インタラクションのシミュレータにおける設計

に保持しておく．そして，経路選択をするときに各車両は，自身が通過時間を保持しているリンクについては，自身の通過時間を予想通過時間とする．自身が通過しなかったリンクの場合は，自身の持つ情報源に，自身が通過する予定の時間帯における指定したリンクの通過時間を要求し，返ってきた通過時間の平均を予想通過時間とする．情報源となる車両は，要求された時間帯に要求されたリンクを通過していた場合，そのリンクの通過時間に補正率を掛けたものを，要求を出した車両に返す．情報源のうちの誰からも通過時間が取得できなかった場合は，各車両は従来と同じ方法で予想通過時間を出す．各車両のリンクの通過時間は，プランの再選択が終わった時点で破棄される．

第4章　シミュレーション内容

本論文におけるシミュレーションでは，各車両が個別に通過時間を取得することができる車両を情報源として定義する．そして，各車両は情報源からの通過時間と，全体の平均の通過時間の2つを併用して経路選択を行う．ここで，情報源からの情報は，全体の平均よりも入手しやすいとして，全体の情報よりも優先して使用される．このような状況において，社会的インタラクションが交通に与える影響を正確に反映していることを確認するために，以下の設定でシミュレーションを行う．

4.1　虚偽の情報を配信する車両の影響

本論文における設定では，大規模なシミュレーションでない限り，各車両の通過時間は全車両の平均の通過時間とほとんど変わらないため，情報源となる車両が通過時間をそのまま伝えても，従来の設定との差はほとんど出ないと考えられる．そこで、他者と社会的インタラクションを行うときの，情報を渡す側の性質について考える．

他者に情報を伝える時は，必ず真実の情報を伝えるとは限らない．中には，自分が利用する経路が混雑するのを防ぐために，実際よりも混雑していると虚偽の情報を流す人も存在し得る．虚偽の情報とは，自分が通過予定の経路を他の車両が選択することを防ぐために，自分の実際の通過時間にいくらかの重みを掛けたものとする．この実験では，虚偽の情報を流す車両がいる場合，交通全体にどのような影響を与えるのか調べる．また，虚偽の情報を流す車両と実際

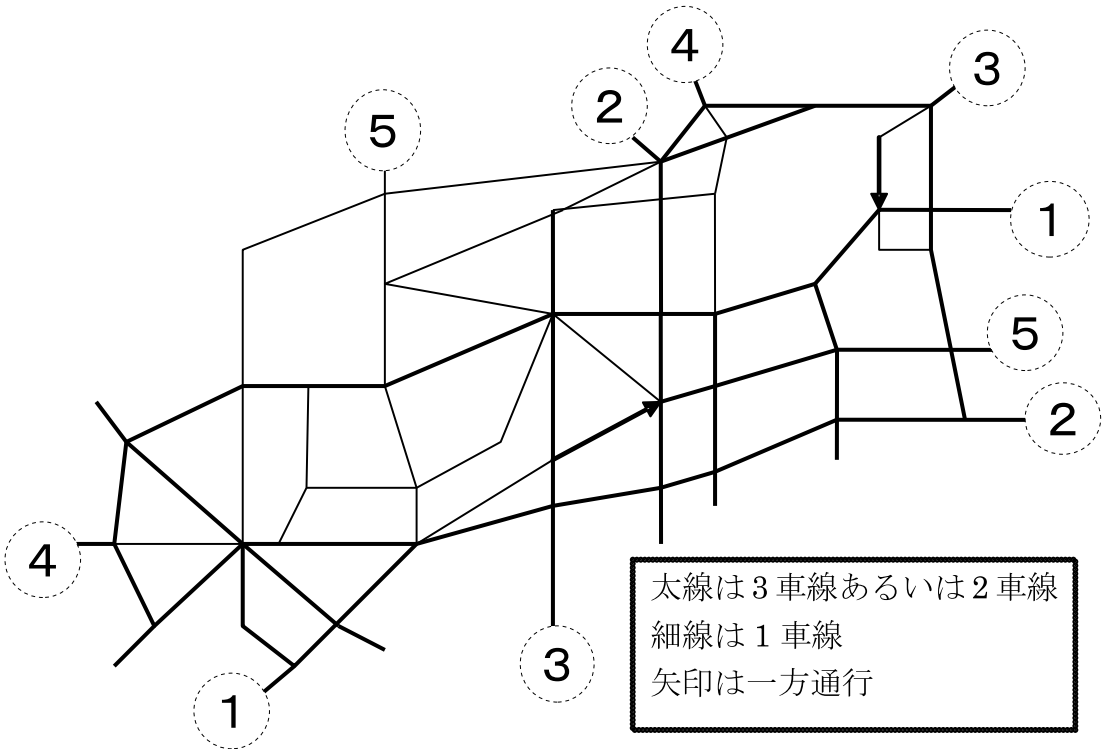

図 6: ネットワークの構造

の情報を流す車両では，旅行時間に差が出るのか調べる．

MATSim の既存の設定では，全車両の平均を元に経路選択をしているため，シミュレーションを繰り返せば，最適な状況に収束する．そのような状況において，虚偽の情報を流すと，各リンクの車両の割り当ての均衡が崩れるため，全体的に効率が下がり，旅行時間も長くなると予想される．また，通過車両数が減少してもリンクの通過時間は希望速度で走ったとき以下にはならないので，虚偽の情報を配信することによる優位性は少ないと考えられる．

4.2 シミュレーション設定

今回の実験における基本的な設定は以下の通りである．

- 現実にある地図を簡略化した，図 6 のネットワークを使用する．
- ネットワーク全体の大きさは約 3km 四方である．
- 各車両は，ネットワークにおける道路を細分化した，リンク上の移動を繰り返して目的地に向かう．
- 車両の希望走行速度 (最高速度) が 8m/s(28.8km/h) の車両と 4m/s(14.4km/h)

の車両がそれぞれ半分ずつ存在する．

- 各車両は，出発地点から，１つだけ決められた目的地への移動を，１日に１度だけ行う．
- 全ての車両は，リンク上を移動している時に，自分の目の前に自分よりも遅い車両がいて，さらに隣の車線が空いているときは，車線変更をして目の前の車両を追い越す．1 車線の車線のリンク上では追い越しはできない．
- １日の移動が終わった後で，プランの評価と選択を行う．
- 全車両の 8 割は，自身が利用したプランの中で，最も評価値が良かったものを選択する．この設定では，経路の移動時間が最も短かったものを選択する．
- 全車両の 2 割は，直前のシミュレーションでの通過時間を基に各リンク予想通過時間を出して，それを利用して経路選択をする．そして，最も予想通過時間の合計が短い経路を持つ新たなプランを作成し，それを選択する．
- 予想通過時間の決定方法は以下のとおりである．
  - 自分が通った経路は，自分が通った時の通過時間を利用する．
  - 情報源となる車両が通ったリンクは，情報源の各リンクの平均通過時間を利用する．
  - 情報源となる車両が 1 台も通らなかったリンクは，全ての車両の平均通過時間を利用する．
  - 全ての車両が通らなかったリンクは，約 30km/h の一定速度で走った時の通過時間を利用する．
- 以上のシミュレーションを 50 日 (iteration) 分くり返す．
- 800 台の車両が同時に出発する．ただし，実際には各車両は一定の間隔をおいて 1 台ずつ移動を始める．
- 図 6 における 1 から 5 の数字のうち，同じ数字を持つ地点の組を，出発地点と目的地の組とする．
- 出発地点と目的地の組は，出発地点と目的地を入れ替えたものも含めた，上記の 10 種類から車両ごとにランダムで決まる．
- 情報源の数は車両ごとに 100 台で，ランダムに決定される．
- 虚偽の情報を流す車両は，直前に走行したときの通過時間に以下のような修正をする．
  - 自分の持つプランから評価値が最高のものを選択する場合で，次回に

使用するプランにおいて通過する予定のリンクである場合，そのリンクの通過時間を実際の 2 倍にして他の車両に伝えることで，他の車両が同じ経路を使用しないようにする．

  - 次回に使用するプランにおいて通過する予定のリンクではない場合や，経路選択をして新たな経路を選択する場合は，通過時間をそのまま配信する．

- 実際の情報を配信する車両は，直前に走行したときの通過時間をそのまま提供する．
- まずは，虚偽の情報を配信する車両が全くいない状況でシミュレーションする．
- 次に，各車両の出発地点，目的地及び情報源を変えずに，全車両中の 10 %，30 %，50 %，70 %，90 %が虚偽の情報を配信する状況でシミュレーションする．
- 各車両の旅行時間は，出発地点において wait2link イベントが発生してから arrival イベントが発生するまでの時間とする．つまり，実際に移動を開始してから目的地に到着するまでの時間が旅行時間となる．
- 虚偽の情報を配信する車両の割合を変えても，各車両の出発地点，目的地及び情報源は変化しないものとする．つまり，情報源から取得する通過時間だけが，各車両の旅行効率に影響する．

4.3 シミュレーション結果

虚偽の情報を配信する車両の割合ごとに全車両の平均旅行時間の変化を表したのが図 7 である．ここで，虚偽の情報を配信する車両を虚偽申告車両とする．図 7 より，虚偽申告車両が多いほど，旅行時間が長くなっているのが分かる．各車両の出発地点と目的地の組，及び情報源を変えて何度かシミュレーションを行っても，同じ傾向が見られた．

旅行時間の減少と旅行効率の改善の関係は，各車両の OD(出発地点，出発時刻，及び目的地) によって変化する．しかし，この実験では，虚偽申告車両の割合を変化させても，各車両の OD は変化しないため，図 7 の結果から，虚偽の情報によって旅行効率が減少していると言える． さらに，虚偽の情報を配信することに優位性があるかどうか調べるために，虚偽申告車両がいる時のシミュレーションにおいて，全車両を 2 つのグループに分けた．虚偽の情報を配信す

る車両をグループ 1，実際の情報を配信する車両をグループ 2 として，それぞれのグループに所属する車両の平均旅行時間を取得した．

次に，虚偽申告車両がいない時のシミュレーションにおいて，全車両を虚偽申告車両がいるときのシミュレーションと同じグループ 1，グループ 2 に分けて，それぞれのグループに所属する車両の平均旅行時間を取得した．

この設定では，虚偽申告車両の割合とグループごとに，そのグループに所属する車両の OD が変化するので，虚偽申告車両の割合ごとに，虚偽申告車両の有無によるグループ 1 とグループ 2 の平均通過時間の変化を比較した．その結果を図 8 から図 12 に示す．また，虚偽申告車両の割合ごとに，グループ 1 及びグループ 2 の平均旅行時間の，旅行時間の変動が安定する 10 日目以降の平均を表 1，2 に示す．

この結果を見ると，虚偽の情報を配信する車両がいる場合は，配信する情報の真偽に関係なく，旅行時間が長くなっていることが分かる．そして，虚偽の情報を配信する車両が多いほど，全ての車両が実際の情報を配信する時との平均旅行時間の差が大きくなっている．以上の結果より，虚偽の情報を流すことが，交通全体の効率を下げることがシミュレーションによって示された．

実際の情報を配信する車両の移動時間だけでなく，虚偽の情報を配信する車

図 7: 虚偽の申告をする車両の割合による平均旅行時間の変化

両の移動時間も長くなったのは，自分が通るリンクの通過車両数を減らした時の移動時間の減少の効果よりも，他の車両が通るリンクの通過車両数を増やした時の移動時間の増加の効果の方が大きいためと思われる．

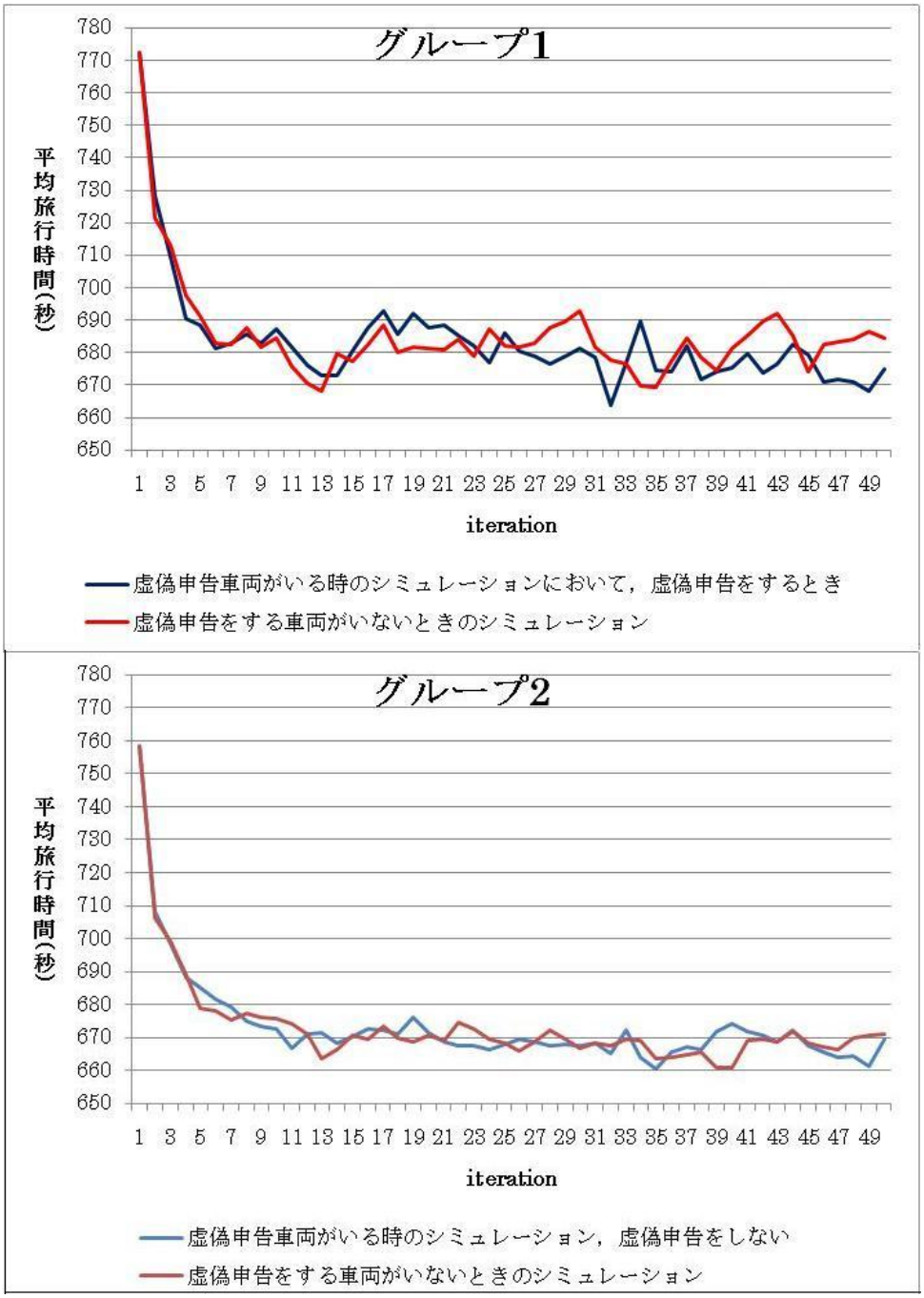

図 8: グループ 1 に所属する車両が全体の 10%のとき

第5章　関連研究

外部情報の入手方法がもたらす影響に関するシミュレーションを用いた研究は，数多くなされてきている．それらの研究の中から，交通における経路ナビ

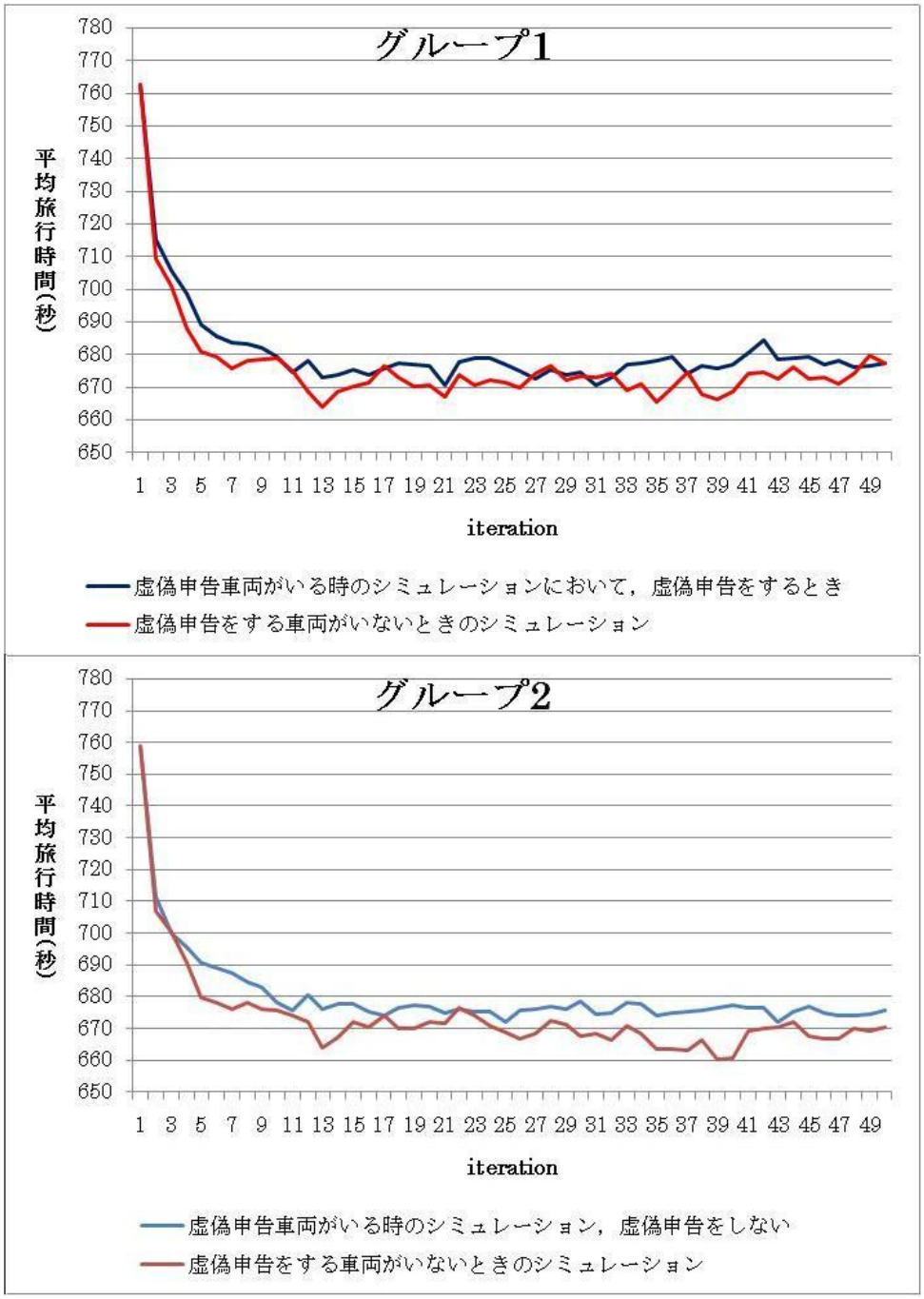

図 9: グループ 1 に所属する車両が全体の 30%のとき

ゲーションの問題に関するものと，テーマパークにおける待ち時間の解消の問題に関するものについて説明する．この 2 つは，各エージェントの経路やスケジュールをコントロールすることで，動的に変化する資源を効率よく割り当てることを目的としている．

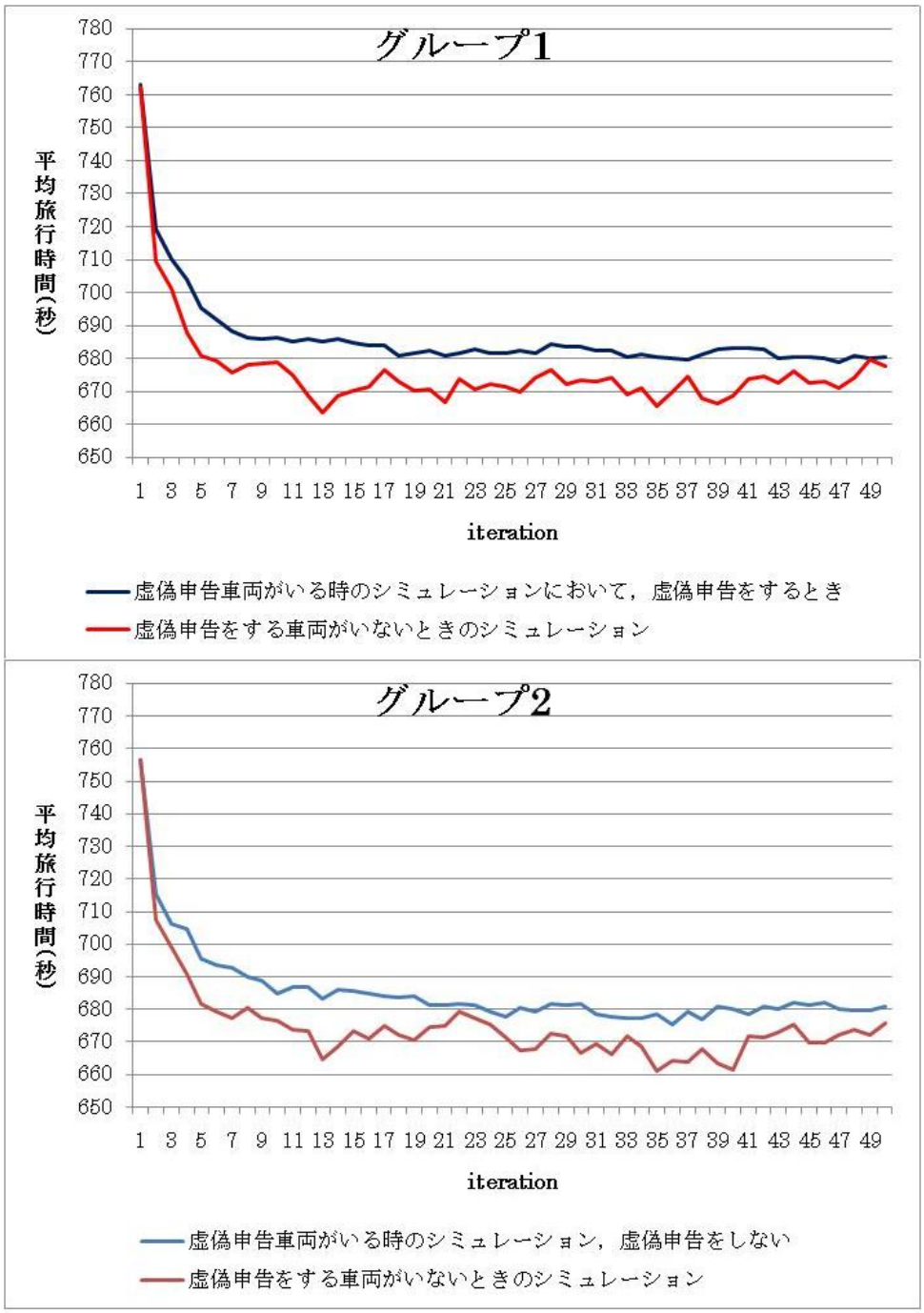

図 10: グループ 1 に所属する車両が全体の 50%のとき

5.1　経路ナビゲーション問題

道路ネットワーク上を多数の車両が通過する時に，各車両の経路選択をコントロールすることで，システムの利用者と全体の移動効率を改善する方法がいくつか提案されている．

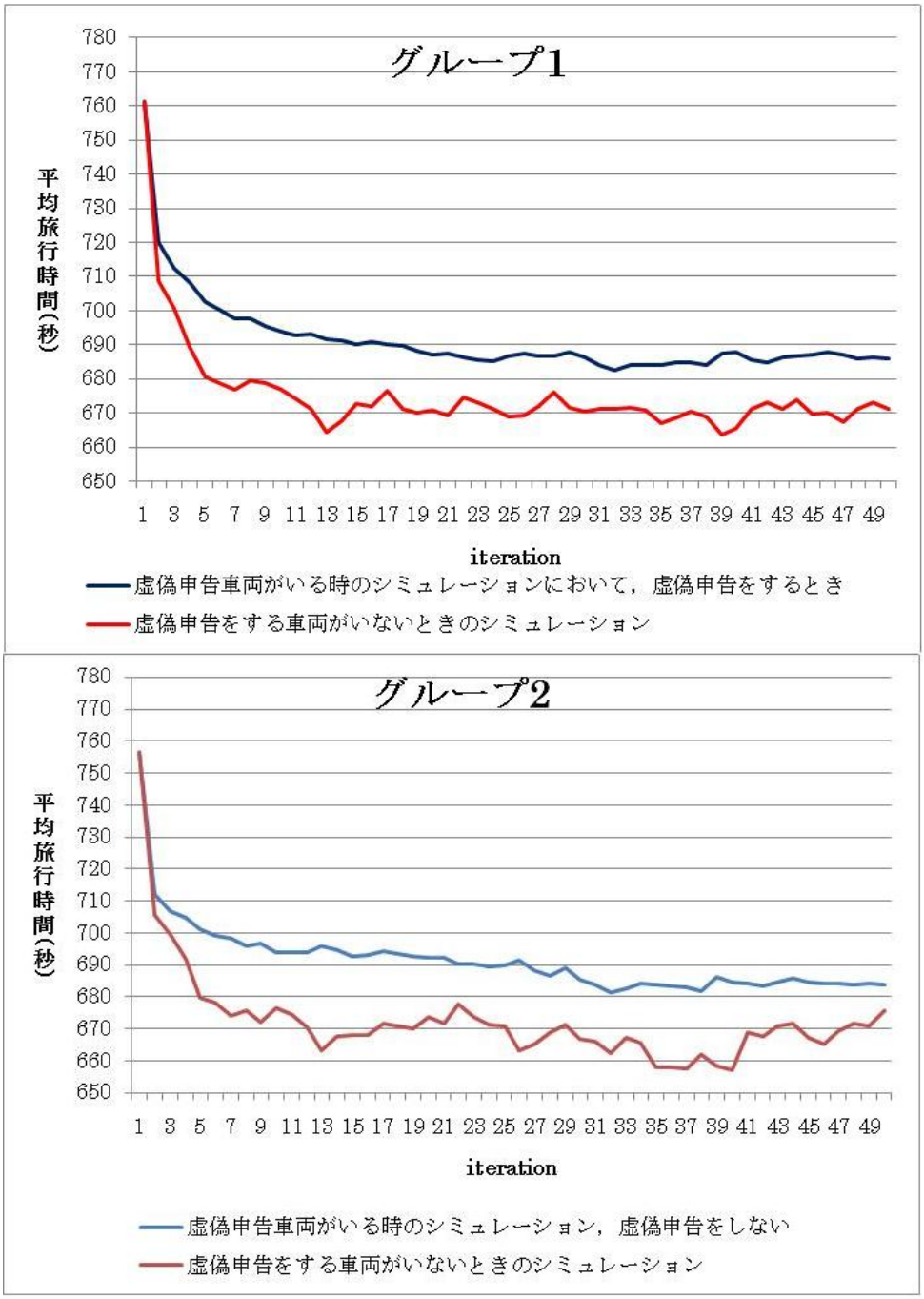

図 11: グループ 1 に所属する車両が全体の 70%のとき

大口らの研究では，一対の起終点に対して，経路が左右2つに分岐し，再度合流しているような道路において，電光掲示板で各車両に両方の経路の現在の混雑状況の情報を提供した結果，左右どちらかの経路に車両が集中するという現象を交通シミュレーションによって再現している [2]．この道路リンクは現実に

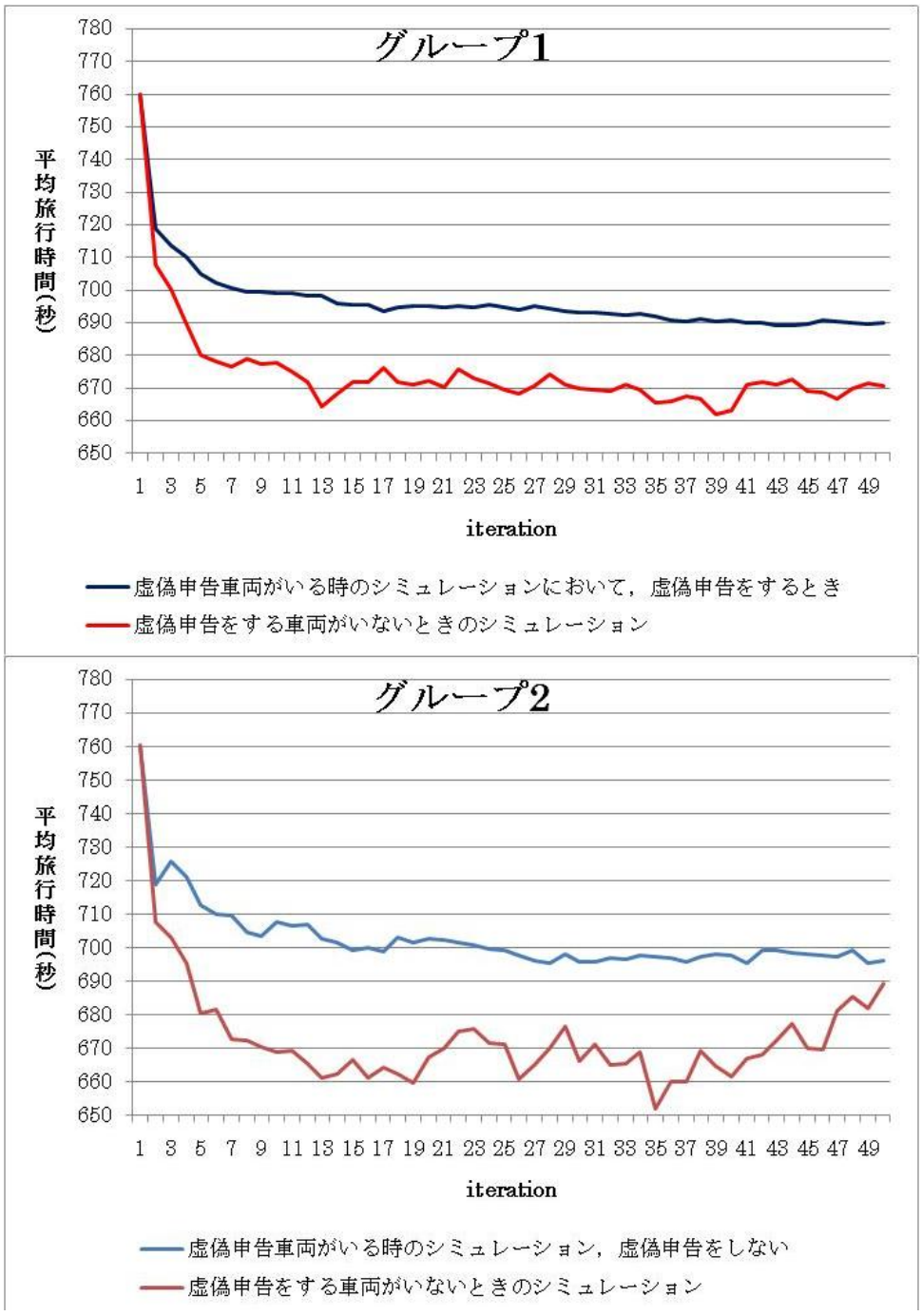

図 12: グループ 1 に所属する車両が全体の 90%のとき

表 1: グループ 1:虚偽の情報を配信する時と通常の情報を配信する時の平均旅行時間の比較

虚偽申告車両の割合 (%)	虚偽の情報を配信する時 (秒)	通常の情報を配信する時 (秒)
10	678.86	681.27
30	676.37	671.85
50	682.08	669.02
70	687.04	670.70
90	692.98	669.97

表 2: グループ 2:虚偽申告車両がいる時といない時の平均旅行時間の比較

虚偽申告車両の割合 (%)	虚偽申告車両がいる時 (秒)	虚偽申告車両がいない時 (秒)
10	668.51	668.54
30	675.89	668.95
50	680.96	670.62
70	687.72	667.77
90	698.83	668.46

存在するものであり，この現象も，交通量が多い時に，現実に起こり得ることが明らかになっていたものである．さらに，大口らは，この現象が確定した将来から見た情報提供の遅れによって起こっていることと，情報の提供を短い周期で中止することが混雑の改善に効果があることを，シミュレーションによって示している．

移動効率の低下につながる，情報提供の遅れを軽減する手段として，各車両の予定走行経路を共有する経路決定戦略が研究されている．山下らは，交通情報センターから配布される現在の交通状況に加えて，各車両の予定走行経路を経路情報サーバに集約し，それらの情報から算出した期待混雑度を各車両に送り，この期待混雑度を元に各車両が経路選択を行うという手法を提案している [3]. 彼らは，常に最短距離経路を選択する戦略，現在の交通状況のみを元にした最短時間経路を選択する戦略，そして経路情報を共有する戦略の 3 つを，マルチ

エージェントシミュレーションによって比較している．その結果，単純な道路網の場合は，経路情報共有戦略によって個人と全体の移動効率が向上したが，放射環状網の場合は，経路情報共有戦略の車両が増えることで，全体の移動効率は向上するものの，個人の移動効率では最短距離経路戦略の方が優れているという，社会的ジレンマ構造が確認されている．

栗山らの研究では，車両ごとに指定したいくつかの目的地を巡回するときに，目的地とその優先度の情報を集計することで，各車両が指定した時間までに出発地点に戻りつつ，優先度の高い地点をできるだけ多く訪問できるようにするスケジューリング手法を提案している [4]．

また，車車間通信によって，他の車両が通過したときの混雑状況を取得し，それを元に経路選択をする方法 [5] [6] [7] が有効であるという研究がある．この手法では，各車両が入手する交通状況の情報に変化を付けることで、同一の情報を利用して経路選択をすることによる車両の集中が起こりにくいようにしている．

大原らの研究では，常に最短距離経路を選択する戦略，自身が通過した時にかかった時間を利用して最短時間経路を選択する戦略，そして車車間通信を利用して，すれ違った車両と通過時間の情報を共有する戦略の 3 種類を，交通シミュレーションによって比較している [6]．その結果，車車間通信を行う車両の割合が少ない時は，他の戦略よりも効率性が良かったが，その割合が大きくなるにつれて，多数の車両が同様の交通情報を共有するために，効率性が低下することが示されている．

これらの方法により，運転者が経路に関する情報を持っていなくても，交通状況を最適化することができる．しかし，リアルタイムな情報収集と経路選択を行うための装置が必要となるため，実現には時間がかかると思われる．

この論文で検証する方法は，他の車両が通過したときの情報を収集して利用するという点では，車車間通信を利用した経路選択方法と似ている．しかし，車車間通信では直前の情報を取得できるのに対し，この論文の方法では，前日の情報を取得している上に，自分と情報源のリンクの通過時刻が同じとは限らないので，情報の正確性は小さいと考えられる．そのため，本論文で設計した環境では，従来の交通シミュレーションよりも効率が悪くなる可能性がある．

5.2 テーマパーク問題

車両交通における経路選択以外でも，外部から情報を提供することが，混雑状況に影響を与える状況がある．例えば，遊園地などの大型集客施設において，多くの人が同じ混雑情報を元に行動した時に，特定のアトラクションに人が集中するという現象が見られる．そのような現象を防ぐために，各エージェントのスケジュールを調整する情報提供の手法が研究されている．

ここで，テーマパーク問題とは，複数のアトラクションからなるテーマパーク内を多数のエージェントが訪問するときに，スケジュールの調整によって，エージェント全体の満足度の向上を目指す問題である．川村らの研究では，マルチエージェントシミュレーションを用いて，交雑情報の提供が有効となる状況を調べている [8]．具体的には，混雑状況の情報に従う戦略と，自分の選好度に従う戦略の 2 種類のエージェントを用意し，2 つのアトラクションから 1 つを選択するという状況でシミュレーションを行っている．

その結果としては，まず，エージェントが利用する戦略の割合を変化させた場合，ある割合において全体の待ち時間が最小となることが示されている．この待ち時間が最小となる点を最小点とする．次に，最小点において全体の混雑度を変化させた場合，混雑度が高すぎても低すぎても待ち時間の減少効果が小さくなることが示されている．さらに，この効果が大きくなるタイミングは，アトラクションの稼働率と関係があることも示されている．そして，混雑情報の提示からアトラクションに着くまでの時間遅れを変化させた場合，最小点では待ち時間はほとんど変化しなかったが，全エージェントが混雑情報に従う場合，時間遅れが大きいほど待ち時間も長くなることが示されている．

鈴木らは，イベント会場における混雑情報の提供が来場者の作る混雑に及ぼす影響を，マルチエージェントシミュレーションによって調べている [9]．その準備として，複数の施設が設置されたイベント会場において．多数の来場者が，自分が見たいと思う施設に向かって移動するというモデルを構築し，まずは施設の配置の違いが混雑状況に与える影響を調べている．その結果，会場の中心付近にある施設ほど混雑しやすく，会場の端にある施設は，シミュレーション終盤に人が集中する傾向があることが示されている．

次に，各施設を出る時に，一定の確率で混雑情報を受け取ることができるものとしてシミュレーションを行っている．その結果，情報提供確率がある値より

小さい時は，情報提供確率が大きくなるにつれて待機エージェント数の偏りが減少しているが，情報提供確率が大きすぎる時は，待機エージェント数が大きく振動することを明らかにしている．さらに，来場者数を増やして，長い時間シミュレーションを行っていると，全ての施設が混雑した状態になるため，情報提供によるエージェントの分散の効果が減少し，その結果，ある１つの施設の待機エージェント数が突発的に大きく増加するという現象が起こることを明らかにしている．

第6章　おわりに

本論文では，社会的なインタラクションの表現方法として，他人との情報交換によって，車両ごとに特定された情報源からの情報を入手するという手法を考えた．そして，この手法によって得られた情報を経路選択に利用することが交通全体に与える影響を調べるためのシミュレーション環境の設計と実装を行った．

本論文では，日常的に目的地に行っているという状況を想定していたが，新しい施設ができたときのシミュレーションなど，利用者の多くが初めて目的地に行く場合でも，今回の手法は利用可能である．なぜならば，自分はまだ目的地に行ったことが無くても，その場所に行く予定があることが分かっているならば，他の人から情報を集めることはできるからである．

また，この論文での交通シミュレーションでは，入手した情報が１日に１回しか反映されなかったが，通信システムを利用してリアルタイムに情報を入手できるような場合でも，今回のような手法が利用できる．なぜなら、例えば，システムの加入者全員が，現在の位置情報や速度を他の加入者に知らせるシステムの場合，プライバシーの保護のために，限られた加入者にのみ情報を送るようにしたいという人がいるかもしれないし，自分と同じような運転行動の人の情報のみを利用したいという人もいるかもしれない．以上のように，情報源を限定した時のシステムの効率の変化を調べるのが，今後の課題である．

謝辞

本論文に対して助言をして頂いた，石田亨教授，松原繁夫准教授，服部宏充助教，中島悠研究員，及び石田・松原研究室の皆様に，深甚の謝意を表する．

参考文献

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

[2] 大口敬, 佐藤貴行, 鹿田成則: 渋滞時の代替経路選択行動に与える交通情報提供効果, 席エ - 土木計画学研究・論文集 (2005).

[3] 山下倫央, 和泉潔, 車谷浩一: 交通流における経路情報の共有に基づいた経路選択の効果の検証, 情報処理学会研究報告. ICS, [知能と複雑系], pp. 71–76 (2004).

[4] 栗山恭嘉, 村田佳洋, 柴田直樹, 安本慶一, 伊藤実: 将来の混雑状況予測に基づく混雑回避巡回スケジューリング手法の提案, 情報処理学会研究報告. ITS, [高度交通システム], pp. 63–70 (2007).

[5] 仲村幹子, 遠藤聡志, 山田孝治, 當間愛晃, 赤嶺有平, 稲葉江: 渋滞緩和を想定した車車間通信による交通情報共有に関する研究, 第 16 回インテリジェントシステム・シンポジウム講演論文集, pp. 167–170 (2006).

[6] 大原健, 能島裕介, 石渕久生: 交通渋滞解消のための車車間通信を用いた道路情報共有の有効性, 第 23 回ファジィシステムシンポジウム講演論文集, pp. 771–774 (2007).

[7] Eichler, S., Ostermaier, B., Schroth, C. and Kosch, T.: Simulation of Car-to-Car Messaging: Analyzing the Impact on Road Traffic, *13th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (2005).

[8] 片岡崇, 川村秀憲, 車谷浩一, 大内東: テーマパーク問題における混雑状況の提示とその効果, 情報処理学会研究報告. ICS, [知能と複雑系], pp. 77–82 (2004).

[9] 鈴木麗璽, 有田隆也: 行動多様性に対する情報共有の影響とその適応性 : イベント会場における混雑情報提供に関するマルチエージェントシミュレーション, 電子情報通信学会論文誌, Vol. J86-D-1, pp. 830–837 (2003).

付録：プログラム

本論文において使用されたMATSimの，自らが行った追加と変更の内容及びソースコードを表記する．ソースコードはjavaで書かれている．

A.1 populationパッケージ

エージェントに関連する情報を取得，保持するためのクラスを含むパッケージ

A.1.1 PersonFriendsImpleクラス

エージェントの情報を格納するクラス．従来のMATSimに存在するPersonImpl, BasicPersonImplの属性に，個別の通過時間を入手可能な車両のIDのリスト(friends)，グループ番号(shareGroup, この番号が同じエージェント間は，通過時間の情報を交換できる)，情報を渡す時の補正値(lieRate)，情報を受け取る時の補正値(shareRate)を追加した．

A.1.2 PopulationReaderMATSimV4クラス

各エージェントの特性と1日の行動計画を記録しているplans.xmlファイルから情報を読み取るためのクラス．前述の情報源に関する情報を取得するためのメソッドを作った．以下にそのコードを示す．

```
private void startFriends(final Attributes atts) {
  if (this.currperson instanceof PersonFriendsImpl){
    String groupString = atts.getValue("group");
    int group = 0;
    if (groupString != null)
      group = Integer.parseInt(groupString);
    String shareRateString = atts.getValue("share_rate");
    double shareRate = 1.0;
    if(shareRateString != null)
      shareRate = Double.valueOf(shareRateString);
    String lieRateString = atts.getValue("lie_rate");
    double lieRate = 1.0;
    if(lieRateString != null)
      lieRate = Double.valueOf(lieRateString);
    if(atts.getValue("list") != null)
```

```
      ((PersonFriendsImpl) this.currperson).setFriends(atts.getValue("list"));
    ((PersonFriendsImpl) this.currperson).setShareGroup(group);
    ((PersonFriendsImpl)  this.currperson).setShareRate(shareRate);

    ((PersonFriendsImpl) this.currperson).setLieRate(lieRate);
  }
}
```

A.2 trafficmonitoring パッケージ

リンクの旅行時間の取得と保持に関連するクラスを含むパッケージ

A.2.1 TravelTimeCalculator クラス

交通シミュレーション中のリンクの通過履歴を集計し，経路選択時にその履歴を提供するクラス．変更点は以下の通りである．

従来からある，各リンクの ID と履歴の組を格納する linkData に加えて，エージェントとリンクの ID と履歴の組を格納する personLinkTravelTime を追加した．また，情報共有するグループごとに linkData を作成するようにした．各車両は，リンクを出る時に，全体の linkData と自分のグループの linkData あるいは personLinkTravelTime に通過時間を入れる．以下にコードを示す．

```
public void handleEvent(final BasicLinkLeaveEvent e) { //リンクを出たときの操作
  if (this.calculateLinkTravelTimes) {
    BasicLinkEnterEvent oldEvent = this.linkEnterEvents.get(e.getPersonId());
    //直前にリンクに入った時のイベント
    if (oldEvent!= null) {
      DataContainer container = getDataContainer(0, e.getLinkId(), true);
      //全車両の通過時間を集計
      this.aggregator.addTravelTime(container.ttData, oldEvent.getTime(),
 e.getTime());
      //通過時間を足し合わせて，通過車両数を 1 増やす
      container.needsConsolidation = true;
      Person person = this.population.getPersons().get(e.getPersonId());
      if(person instanceof PersonFriendsImpl) {
```

```
        double group = ((PersonFriendsImpl)person).getShareGroup();
        if(group > 0) {
          container = getDataContainer(group, e.getLinkId(), true);
          //グループごとの通過時間を集計
          this.aggregator.addTravelTime(container.ttData, oldEvent.
getTime(), e.getTime());
           container.needsConsolidation = true;
        } else {
           TravelTimeData data = getTravelTimeData(e.getPersonId()
, e.getLinkId(), true);
           this.aggregator.setTravelTime(data, oldEvent.getTime()
, e.getTime());
          //車両ごとに通過時間を記録する.
          }
        }
      }
    }
  }
}

public void handleEvent(final BasicAgentArrivalEvent event) {
  /* remove EnterEvents from list when an agent arrives.
   * otherwise, the activity duration would counted as travel
 time, when the
   * agent departs again and leaves the link! */
  this.linkEnterEvents.remove(event.getPersonId());
}

public void handleEvent(BasicAgentStuckEvent event) {
  //途中でシミュレーションが終わったとき
  BasicLinkEnterEvent e = this.linkEnterEvents.remove(event.
getPersonId());
```

```
  if (e != null) {
    TravelTimeData data = getTravelTimeData(e.getPersonId(),
 e.getLinkId(), true);
    this.aggregator.addStuckEventTravelTime(data, e.getTime()
, event.getTime());
    //リンクに入ってからシミュレーション終了までの時間を記録する.
    if (this.calculateLinkToLinkTravelTimes){
      log.error(ERROR_STUCK_AND_LINKTOLINK);
      throw new IllegalStateException(ERROR_STUCK_AND_LINKTOLINK);
    }
  }
}

private TravelTimeData getTravelTimeData(final Id personId, final
 Id linkId, final boolean createIfMissing) {
  TravelTimeData data = this.personLinkTravelTime.get(new Tuple<I
d, Id>(personId, linkId));//データエントリーの取得
  if ((null == data) && createIfMissing) {
    //データエントリーが無ければ作成する
    data = this.ttDataFactory.createTravelTimeData(linkId);
    this.personLinkTravelTime.put(new Tuple<Id, Id>(personId,
linkId), data);
  }
return data;
}
private DataContainer getDataContainer(double group, final Id
linkId, final boolean createIfMissing) {
  DataContainer data = this.linkData.get(new Tuple<Double, Id>
(group, linkId));
  //データエントリーの取得
  if ((null == data) && createIfMissing) {
    //データエントリーが無ければ作成する
```

```
    data = new DataContainer(this.ttDataFactory.
createTravelTimeData(linkId));
    this.linkData.put(new Tuple<Double, Id>(group,
linkId), data);
  }
  return data;
}
```

リンクの通過時間を取得するときは，優先度の低い情報を優先度の高い情報で上書きするようにした．また，他の車両からの個別の通過時間への補正も適用する．以下に通過時間を取得するメソッドのコードと，通過時間をリセットするときのコードを示す．

```
public double getLinkTravelTime(final Person person,
final Link link, final double time) {
  if (this.calculateLinkTravelTimes) {
  //全体の平均通過時間を取得
  DataContainer container = getDataContainer(0, link.
getId(), true);
    if (container.needsConsolidation) {
      consolidateData(container);
    }
    double travelTime = this.aggregator.
getTravelTime(container.ttData, time);
    TravelTimeData data;
    if(person instanceof PersonFriendsImpl) {
      double group = ((PersonFriendsImpl)person).
getShareGroup();
      if(group > 0) {
//グループごとの平均通過時間を取得
        container = getDataContainer(group,
link.getId(), false);
if(container != null) {
  if (container.needsConsolidation) {
```

```
    consolidateData(container);
  }
        travelTime = this.aggregator.
getTravelTime(container.ttData, time);
}
      } else {
        //情報源の通過時間を収集
        double sum = 0.0;
int cnt = 0;
for(Id friendId : ((PersonFriendsImpl)
person).getFriends()) {
  data = getTravelTimeData(friendId,
link.getId(), false);
  if (data != null) {
    double ftime = this.aggregator
.getTravelTime(data, time);
    Person friend = this.population.
getPersons().get(friendId);
    if(friend instanceof
 PersonFriendsImpl) {
      if(!((PersonFriendsImpl)friend).
getChangeRoute()) {
if(passNextTime(friend, link.
getId())){
                //他人に通過時間を渡す時の補正
  ftime = ftime*((PersonFriends
Impl)friend).getLieRate();
}
      }
    }
          //add friends' travel time if they
 passed this link
```

```
    sum += ftime*((PersonFriendsImpl)
person).getShareRate();
    cnt++;
  }
}
if (cnt != 0) {
  travelTime = sum / cnt;
}
      }
      data = getTravelTimeData(person.getId(),
link.getId(), false);
      if(data != null) {
travelTime = this.aggregator.
getTravelTime(data, time);
      }
    }
  return travelTime;
  }
throw new IllegalStateException("No link travel time
 is available " +"if calculation is switched off
 by config option!");
}

public void reset(int iteration) {
//プランの再決定後にリセットする
  if (this.calculateLinkTravelTimes) {
    for (Tuple<Id, Id> key : this.
personLinkTravelTime.keySet()){
      Person person = this.population.
getPersons().get(key.getFirst());
      Id linkId = key.getSecond();
```

```
      TravelTimeData data = this.
personLinkTravelTime.get(key);
        if(passNextTime(person,
 linkId)) {
        data.resetTravelTimes();
//次回通過予定のリンクは残す
      } else {
this.personLinkTravelTime.remove(key);
      }
    }
    for (DataContainer data : this.
linkData.values()){
      data.ttData.resetTravelTimes();
      data.needsConsolidation = false;
    }
  }
  if (this.calculateLinkToLinkTravelTimes){
//リンク間の通過時間, 使用しない
    for (DataContainer data :
this.linkToLinkData.values()){
    data.ttData.resetTravelTimes();
    data.needsConsolidation = false;
    }
  this.personLinkToLinkTravelTime.clear();
  }
this.linkEnterEvents.clear();//リンクに残っている車両を消す
}

private boolean passNextTime(Person p, Id linkId) {
  //エージェントが選択したプランの経路は, 指定したリンクを通るか調べる
  Route route = null;
  for (PlanElement pe : p.getSelectedPlan().getPlanElements()) {
```

```
    if (pe instanceof Leg) {
      route = ((Leg) pe).getRoute();
      if(route instanceof NetworkRoute) {
if (((NetworkRoute) route).getStartLinkId().equals(linkId)
|| ((NetworkRoute) route).getEndLinkId().equals(linkId)) {
  return true;
}
if (((NetworkRoute) route).getLinkIds().contains(linkId)) {
  return true;
}
      }
    }
  }
return false;
}
```

A.3 replanning パッケージ

1 日の行動計画であるプランの再選択のためのクラスのパッケージ

A.3.1 StrategyManager クラス

プラン変更戦術を選択し，プラン変更を行うクラス.

各車両が経路探索をするとき，選択した経路を通過するかどうか調べるために，全車両がプランの選択をした後で，経路変更を行うようにした．以下にメインのコードを示す.

```
public void run(final Population population) {
  this.personStrategies = new HashMap<Id, PlanStrategy>
(population.getPersons().size());
  // initialize all strategies
  for (PlanStrategy strategy : this.strategies) {
    strategy.init();
  }
  // then go through the population and assign each person to a strategy
```

```
  for (Person person : population.getPersons().values()) {
    if ((this.maxPlansPerAgent > 0) && (person.getPlans().size()
 > this.maxPlansPerAgent)) {
      removePlans(person, this.maxPlansPerAgent);
    }
  PlanStrategy strategy = this.chooseStrategy();
  if (strategy != null) {
    this.personStrategies.put(person.getId(), strategy);
    strategy.run(person, 1);
  } else {
    Gbl.errorMsg("No strategy found!");
    }
  }
  // if a person changed a plan, assign to a strategy one more time
  for (Person person : population.getPersons().values()) {
    PlanStrategy strategy = this.personStrategies.get(person.getId());
    strategy.run(person, 2);
  }
  // finally make sure all strategies have finished there work
  for (PlanStrategy strategy : this.strategies) {
    strategy.finish();
  }
}
```

A.3.2 StrategyManager クラス

StrategyManager から呼び出され，プランの選択と経路探索を行うクラス．

前述の StrategyManager の変更に合わせて，プラン選択と経路探索の処理を分離した．また，経路探索を行うかどうかの情報を PersonFriendsImpl に渡すようにした．以下にメインのコードを示す．

```
  public void run(final Person person, final int times) {
  if(times == 1) {//プラン選択
    this.counter++;
    Plan plan = person.getRandomUnscoredPlan();
```

```
    if (plan == null) {
      plan = this.planSelector.selectPlan(person);
    }
    person.setSelectedPlan(plan);
    if (this.firstModule != null) {//モジュールを持っている場合は，経
路探索を行う
      plan = person.copySelectedPlan();
      this.plans.add(plan);
      ((PersonFriendsImpl)person).setChangeRoute(true);
    } else {
      ((PersonFriendsImpl)person).setChangeRoute(false);
    }
  } else if (times == 2) {//経路探索
    Plan plan = person.getSelectedPlan();
    // start working on this plan already
    if (this.firstModule != null) {
      if (this.firstModule instanceof PlanStrategyModule) {
        ((PlanStrategyModule) this.firstModule).handlePlan(plan);
      } else if (this.firstModule instanceof BasicPlanStrategyModule) {
        ((BasicPlanStrategyModule) this.firstModule).handlePlan(plan);
      }
    }
  }
}
```