

モバイルワイヤレスネットワークシミュレータ MobiREAL の並列実行による大規模シミュレーションの検討

小西一樹[†] 廣森聡仁[†] 山口弘純[†] 安本慶一[‡] 東野輝夫[†] 谷口健一[†]

大阪大学 大学院情報科学研究科[†]
奈良先端科学技術大学院大学 情報科学研究科[‡]

A Study on Parallel Execution of MobiREAL Wireless Network Simulator

Kazuki Konishi[†], Akihito Hiromori[†], Hirozumi Yamaguchi[†],
Keiichi Yasumoto[‡], Teruo Higashino[†], Kenichi Taniguchi[†]

Graduate School of Information Science and Technology, Osaka University[†]
Graduate School of Information Science, Nara Institute of Science and Technology[‡]

1 はじめに

近年、無線技術の発展により、移動無線端末(ノード)が無線を用いて互いに通信しあうようなモバイルワイヤレスネットワーク上のネットワークプロトコルや通信システムの開発が盛んに行われている。数千、数万ノード規模のアプリケーションやプロトコルの性能評価にはネットワークシミュレーションが不可欠であるが、ns-2 や GloMoSim などの既存のネットワークシミュレータでは、ノードのモビリティモデルは単純であることが多く、評価対象とするアプリケーションや、その評価内容によっては、この単純化により評価の精確性を大きく損なう場合がある。例えば、近隣の携帯端末やショートレンジの安価な基地局を介したホップバイホップ通信で端末間のリアルタイム通信を行うというアプリケーションを想定し、AODV や DSR などのルーティングプロトコルの適性を経路切断率により評価する場合を考える。この時、モビリティモデルにランダムウェイポイントを用いて、既存のネットワークシミュレータを利用しシミュレーションを行うと、経路切断率はランダム性によりシミュレーション領域内でほぼ一定であることが予想される。一方、実世界では歩行流という人の流れが存在し、その流れに沿って経路が確立された場合、経路を維持する中継ノードは歩行流に沿う方向に移動するため、経路切断率は低くなると考えられるが、逆に歩行流に直交する経路であれば経路切断率は高くなると考えられる。このことから、シミュレーションにより現実世界での性能をより精確に評価したい場合、ノードのモビリティをより現実世界での人間の行動に近づけることが必要である。

そこで、我々の研究グループでは、ノードの現実に即した行動を表現可能なモバイルワイヤレスネットワーク向けシミュレータ MobiREAL を開発している [1][2]。MobiREAL はノードの行動をシミュレートする行動シミュレータ部、シミュレーション結果を視覚的に表示するアニメータ部、アプリケーション層以下をシミュレートするネットワークシミュレータ部から構成されており、現実的なモビリティを考慮した、大規模モバイルワイヤレスネットワークのシミュレーションを目標としている。

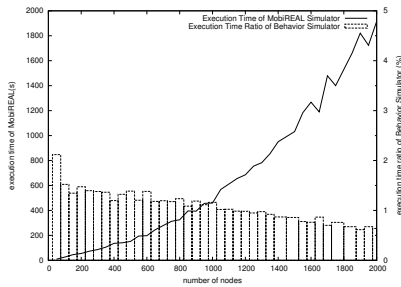
しかし、MobiREAL で、数千、数万ノード規模のシミュレ

ーションを実現するためには、行動シミュレータ部と比較しより負荷の大きいネットワークシミュレータ部の大規模適応性が不可欠である。MobiREAL ネットワークシミュレータ部は、米 Georgia 工科大学で有線ネットワークを対象とした大規模適応性に重点をおいて設計、開発されたネットワークシミュレータ GTNetS [3] に基づき開発され、行動シミュレータ部から定期的に得られるノード位置を用いてシミュレーションを実行する。本稿では、GTNetS に標準で実装されている有線ネットワークの並列実行環境を利用し、無線ネットワークシミュレーションの並列実行を実現するための課題とその解決法を述べる。GTNetS はシミュレーション領域を分割し、各プロセスに分割領域のシミュレーションを担当させることで、有線ネットワークシミュレーションの並列実行を実現している。この方法をモバイルワイヤレスネットワークシミュレーションに適用する場合、有線の場合に比べて、無線伝播計算や、ノードの移動処理など、プロセス間で処理する必要があるイベントが多くなり、そのイベントの処理や、プロセス間メッセージ交換により並列実行の効率が低下してしまう。本稿ではノードの移動特性を利用し、この問題を解決するプロセスへの領域割り当て手法を提案する。

2 研究動機

図 1 は HP の Workstation(CPU Intel Xeon 3.06GHz, メモリ 2GB) を用いて、同図に記載されたシナリオのシミュレーションを行った場合の、ノード数に対するシミュレーション実行時間(左軸)と、そのうち行動シミュレータが占める実行時間の割合(右軸)を表している。この図から、全体のシミュレーション時間において行動シミュレータ部の実行時間が占める割合はわずか数%以下であることが分かる。よって、MobiREAL で大規模無線ネットワークシミュレーションを行うためには、ネットワークシミュレータ部の実行速度改善が重要となる。

速度改善において最も一般的な方法が並列化であり、研究グループでは、GTNetS の並列化機能 [4] を利用した無線ネットワークシミュレーションの並列実行の実現を目標としている。



シミュレーション環境	
シミュレーション領域	500(m) × 500(m)
モビリティモデル	汎用歩行者モデル [2]
アプリケーション	CBR(10Kbps) フロー ×1
ルーティングプロトコル	DSR
MAC 層プロトコル	IEEE 802.11(CSMA/CA with RTS/CTS)
通信距離	300(m)
シミュレーション時間	480(s)

図 1: ノード数 vs 実行時間

3 関連研究と並列実行の問題点

ネットワークシミュレーションの並列実行においては各プロセスへのシミュレーション対象の割り当てが重要である。直観的には、シミュレーション領域を分割し、各プロセスが分割された領域のシミュレーションを担当するという方法がもっとも単純であり、かつ効果が高いと考えられる。

しかし、無線ネットワークのシミュレーションでは、指向性を持たない無線電波の性質により、送信ノードによるパケット送信が、その無線範囲内における全てのノードに影響する可能性がある。そのため、同じプロセスに割り当てられたノード間の通信であっても、無線範囲に含まれる異なるプロセスに割り当てられたノードに対して電波干渉判定などの処理が必要となる。また、無線ネットワーク、特に MobiREAL が対象としているようなモバイルアドホックネットワークではノードが頻繁に移動するため、担当領域の境界を越えてノードが移動する状況が頻繁に起こり得る。

これらの原因により、無線ネットワークシミュレーションでは、有線ネットワークのシミュレーションに比べて担当領域の境界を越えて影響するイベント、つまりプロセス間でのメッセージ交換が必要なイベント（これをプロセス間イベントとよぶ）が多く発生する。これが並列実行の効率を低下させることになるため、なるべくプロセス間イベントを減らす分割が望まれる。

現在まで、無線ネットワークシミュレータの並列化に関するさまざまな研究が行われてきた。最も広く知られる無線ネットワークシミュレータ GloMoSim[5] は PARSEC と呼ばれる並列化言語で実装されており、標準で並列シミュレーションの環境を備えている。しかし、GloMoSim の並列実行には分散共有メモリを搭載した特殊なクラスタマシンが必要であり、汎用性という点で問題がある。また、GloMoSim では領域を地理分割して各プロセスに割り当てる手法を提案しているが、ブロック分割、サイクリック分割など単純な方法を採用するに留まっている。

SWiMNet[6] では、ノード移動や携帯端末の接続要求等のイベントを事前計算し、結果を用いてシミュレーションの実行を行う。事前計算と、シミュレーション実行は並列に実行され、また、それら自身も並列実行される。事前計算ではモビリティグループを同じくするノードを各プロセスに均等に配分し、シミュレーション実行においては負荷バランスを考慮したセル分割によるプロセスへの配分を行うという試みがされている。しかし、

SWiMNet は携帯電話ネットワークに特化したシミュレータであることから、端末間の直接通信を含むアドホックネットワークを対象にする MobiREAL とは対象が異なる。

WiPPET[7] は、同様に TeD(Telecommunication Description Language) を用いて実装されている携帯電話ネットワークを対象としたシミュレータである。WiPPET の並列化では、チャンネル分割および領域分割の 2 パターンが提供されている。領域分割では全体の領域を各部分領域上のノード数が同じになるように、プロセス数と同じ個数の部分領域に分割する手法がとられている。

4 並列化 MobiREAL の設計

4.1 GTNetS の並列化機能

GTNetS の並列機能は IEEE 標準の異機種シミュレーション統合規格仕様である HLA (High Level Architecture) を提供する RTI(Run Time Infrastructure) ミドルウェアを用いて実現される。GTNetS ではシミュレーション領域を分割し、各プロセスに部分領域に位置するノードを担当させる手法を用いている。プロセスの担当ではないノードはプロセス内ではゴーストノードとして扱われ、シミュレーションに必要な最低限の情報のみ保持する。シミュレーションコアクラスでは、RTI の API を用いてシミュレーション時間の同期が行われる。また、担当ノードと非担当ノードを接続するリンクがある場合は、そのリンクに対し、パケット送信が行われた場合、API を呼び出すことで、対応するシミュレータへメッセージを送る。本稿では、シミュレータ間の同期、メッセージの交換などには、この機能を利用するため、シミュレータへの担当領域の割り当て方法にのみ焦点をあてる。

4.2 並列化 MobiREAL アーキテクチャ

並列化 ネットワークシミュレータ部のアーキテクチャを図 2 に示す。行動シミュレータ部からはネットワークシミュレータ部の並列機構を隠蔽するために、行動シミュレータ部との情報交換はインタラクションオブジェクトと呼ぶ連携機能と入出力情報の管理を行うオブジェクトを各シミュレータが共有できるように配置する。ある t シミュレーション時間の実行後、インタラクションオブジェクトと行動シミュレータ部が連携し、アプリケーション入力 (A_{IN})、既存ノードの位置 (P)、速度 (V) に加え、ノードの生成、削除情報を受け取り、その t 時間でのネットワークシミュレーションによって得られたアプリケーション出力 (A_{OUT}) を渡す。各シミュレータはシミュレーション時間 t ごとにインタラクションオブジェクトにアクセスし A_{OUT} とノード情報を取得する。つまり、全プロセスは t 時間のシミュレーションごとに同期することになるが、 A_{OUT} が次の t 時間では得られないことが確定している場合は、インタラクションオブジェクトが A_{OUT} の取得を待たずに行動シミュレータ部と情報交換を行う。担当ノードであるか否かにかかわらず、全てのプロセスは全ノードの位置、速度情報を取得する。あるプロセスにおいて、無線パケット送信が行われる場合、送信ノードと受信ノードとの位置関係から、受信ノードにおける影響の有無を決定する。例えば無線範囲にゴーストノードが含まれている場合

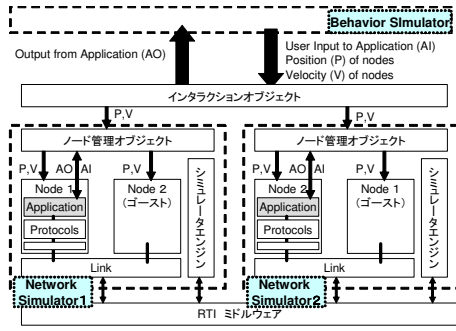


図 2: 並列化 MobiREAL のアーキテクチャ

でも、受信される無線信号の強度が Career Sense Threshold より小さければ、受信ノードでは影響を受けないため、シミュレータ間のメッセージ通信は行われないようにする。

5 領域分割アルゴリズム

並列実行の効率をあげるためには、プロセス間イベントの実行回数を減らすような領域分割手法が必要である。RWP のようなモビリティモデルを用いる場合には、もともと均等なノード配置及びノード移動が期待できるため、領域全体を均等に分割するといった単純な方法でも十分である。しかし、MobiREAL では、ノードはある程度人の流れ（歩行流）が発生するような現実に即したモビリティを持つ。そのため、単純な分割方法では、領域内のノード配置の偏りから、特定のプロセスに負荷が偏ってしまう場合や、歩行流を分断するような分割によりプロセス間イベント数が非常に多くなってしまふ場合がある。

そこで、本稿では MobiREAL の特徴であるノードの現実的なモビリティを利用した効率的な領域分割手法を提案する。MobiREAL では、現実に即したシミュレーションを行うために、グループモビリティ、経路計算、障害物などを採り入れている。単純には、例えば、歩行流に沿って領域を分割すると、歩行流を横断する形で領域を分割した場合に比べ、ノードが分割した領域の境界を越えて移動する回数を減らすことができると予想できる。また、無線を用いたマルチホップ通信のシミュレーションにおいて二点間でリンクが確立された場合も、歩行流に沿う場合はリンクの生存率が高いため、結果的に、歩行流に沿って領域を分割することで、境界をまたぐイベント数を削減できると考えられる。加えて、並列化の効果を上げるためには、各シミュレータに適切に負荷を配分する必要がある。ネットワークシミュレーションに負荷を与える支配的要素はトラフィック量とノード数である。簡単化のためシミュレーションに用いるプロセスの性能が等しいと想定する。このとき、(1) 分割領域の境界を移動するノード数が少ない、(2) 各プロセスが扱うノード数がほぼ等しいという 2 つの条件を満たす分割が望ましいと言える。

我々のアイデアは領域をまず複数のグリッドに分割し、低負荷である行動シミュレータ部のみを実行し、グリッドごとのノードの密度や、グリッド間のノード移動回数を計測する。それを用いて、領域を重みつきグラフに変換し、既存のグラフ分割アルゴリズムを適用して、各プロセスが担当するグリッド集合を求めるといったものである。以降、このアイデアを実現する具体的な手法を示す。

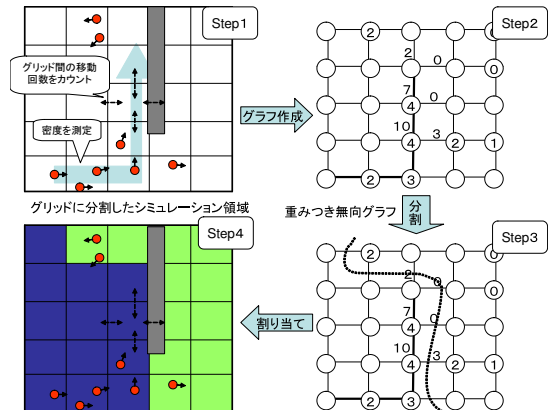


図 3: 提案する領域分割手法

Step0 ノードモビリティシミュレーション 2章で示したように行動シミュレータ部の実行速度はネットワークシミュレータ部に比べて速いため、まず行動シミュレータ部単体でノードモビリティシミュレーションを行う。その結果は以降のプロセスで利用する。

Step1 ノード密度およびグリッド移動回数の計算 まず領域を一辺が $M(m)$ の正方形グリッドに分割する (図 3 Step1)。そして、各グリッドごとに平均ノード密度を、グリッドの境界ごとにノードがグリッドから他のグリッドへ移動する回数を計測する。

Step2 重みつき無向グラフの作成 各グリッドを頂点とし、ノード密度をその頂点の重みとする。また、隣り合うグリッドを表す頂点間を辺で結び、ノードの移動回数をその辺の重みとする。これにより、重みつき頂点および、重みつき辺からなる無向グラフを作成する。

Step3 グラフ分割問題を用いた領域分割 グリッド集合をプロセスに担当領域として割り当てるために、グラフの頂点を k 個のプロセスに分割する。このとき、プロセス間でのノード移動をなるべく少なくするために、異なるプロセスに属する頂点間の辺の重み (ノード移動回数) の和を最小とし、さらにプロセスに割り当てられるノード数をなるべく均一にするために、各プロセスに割り当てられた頂点の重み (平均ノード密度) の総和が各プロセスでほぼ等しくなるようにする必要がある。この問題はよく知られた k -way グラフ分割問題に帰着できる。 k -way グラフ分割問題とは、

$$G = (V, E), V = \{v_1, \dots, v_n\}, E = \{e_1, \dots, e_m\}$$

であるようなグラフに対して分割数 k をあたえたときに V を次の条件を満たす k 個の部分集合に分割する問題である。

$$V = V_1 \cup V_2 \cup \dots \cup V_k, i \neq j \Rightarrow V_i \cap V_j = \emptyset$$

$$W(V_1) = W(V_2) = \dots = W(V_k) (W(V_i) = \sum_{v_j \in V_i} v_j \text{の重み})$$

$$\sum e_i \text{の重み} (e_i \text{は異なる集合に属するノードを結ぶ辺}) \text{が最小}$$

この問題は NP 完全であることが知られているが、いくつかのヒューリスティックアルゴリズムが提案されている。本研究では、Karypis らが提案した Multilevel partition algorithm[8]を用いる。このアルゴリズムはまず、マージアルゴリズムを用いて、 G の頂点と辺を結合して粗い重みつきグラフ G_w を作る操作を再帰的に行う。グラフがある程度小さくなったところで、 k -分割問題、つまり、重みなし無向グラフ G を頂点数を同

ノード数	200	500	1000	2000	5000
Step0(s)	0.86	2.39	6.36	13.76	34.65
グリッドサイズ $5(m) \times 5(m)$					
Step1+2(s)	1.18	1.31	1.53	2.06	3.74
Step3(s)	0.24	0.25	0.25	0.25	0.23
グリッドサイズ $10(m) \times 10(m)$					
Step1+2(s)	0.33	0.45	0.67	1.16	2.75
Step3(s)	0.04	0.05	0.04	0.04	0.05

図 4: (a) ノード数 vs 領域分割計算時間

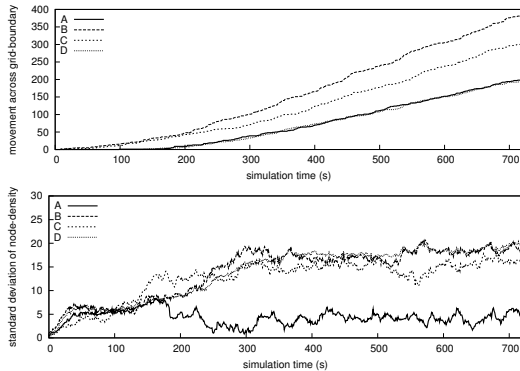


図 4: (b) 領域分割の違いによるプロセスごとのノード数とプロセス間ノード移動数の違い

数, カットエッジ数を最小とするように 2 分割する問題を解く bisection アルゴリズムを繰り返し適用して k 分割を得る. これを初期分割として, もとの G にマッピングし, 部分頂点集合間で Kernighan-Lin の分割アルゴリズムを適用して頂点の入れ替えを行うことでバランスのよい分割に変え, k -way 分割問題の近似解を得るアルゴリズムである. 実装にはグラフ分割問題用のソフトウェアライブラリである METIS[9] を利用した.

Step4 各プロセスへの領域の割り当て Step3 で得られた V_i に相当するグリッド集合をプロセス i の担当領域とすることで, シミュレータ間移動を最小限に抑える領域分割方法を実現できる.

6 評価実験

まず, 提案した領域分割アルゴリズムを C++ で実装し, その実行時間を計測した. 領域サイズを $2000(m) \times 2000(m)$ とし, ノード数を変化させた際の, ノード密度, グリッド間移動回数計測によるグラフ作成 (Step1+Step2) の時間と, そのグラフに対する, k -way グラフ分割問題 ($k = 8$) の求解 (Step3) の時間を図 5(a) に示す. 行動シミュレータ部によるモビリティ計算 (Step0) の時間に比べ, グラフ作成時間, グラフ分割時間は小さく, ネットワークシミュレータ部と連携した場合のシミュレーション時間には, ほとんど影響しないことが推測できる. また, グリッドサイズを大きくすることによりシミュレーション領域内のグリッド総数が減少するためグラフが小さくなり, グラフ分割時間は減少する. したがってある程度大きなシミュレーション領域の場合もグリッドサイズを適当な値に設定すれば, グラフ分割は効率よく行える.

続いて, 提案手法によって求めた領域分割の効率を評価する. $k = 4$ として, モビリティモデルとして歩行流が発生するような, 経路上を人が目的地を目指して移動する汎用歩行者モデルを用

いて, (A) 提案手法で領域分割したものの, (B) 領域を $125(m) \times 500(m)$ の領域に 4 等分したものの, (C) 領域を $500(m) \times 125(m)$ の領域に 4 等分したものの, (D) 領域を $250(m) \times 250(m)$ の領域に 4 等分したものの, の 4 パターンを評価した. 評価項目として, シミュレーション中の各プロセスの担当ノード数の標準偏差と, 領域境界を移動するノード数を測定した. 評価結果を図 5(b) に示す. 提案手法は, 単純に分割したパターンに比べノード数が均等に分担されていることが分かる. また, (B), (C) と比べると, 境界を越えて移動する回数も少なくなっている.

7 おわりに

本稿では, MobiREAL ネットワークシミュレータ部の並列実行を達成するためのアーキテクチャの設計と, MobiREAL の特性を利用した領域分割手法の提案を行った. 今後は, 並列機構の実装を進めていくとともに, 実際の並列環境において提案した領域分割手法の評価実験を行う予定である.

参考文献

- [1] 小西一樹, 内山彰, 廣森聡仁, 山口弘純, 安本慶一, 東野輝夫, 谷口健一. MANET アプリケーション向けのシミュレータ MobiREAL の実装に関する検討. 情報処理学会研究報告 (第 31 回 MBL 研究会), No. 114, pp. 55 – 62, 2004.
- [2] MobiREAL web. <http://www.mobireal.net>.
- [3] G. F. Riley. The Georgia Tech network simulator. In *Proc. of the ACM SIGCOMM Workshop on Models, Methods and Tools for Reproducible Network Research*, pp. 5 – 12, 2003.
- [4] K. S. Perumalla, Alfred Park, R. M. Fujimoto, and G. F. Riley. Scalable RTI-based parallel simulation of networks. In *Proc. of the 17th ACM Workshop on Parallel and Distributed Simulation (PADS'03)*, 2003.
- [5] X. Zeng, R. Bagrodia, and Gerla M. GloMoSim: A library for the parallel simulation of large-scale wireless networks. In *Proc. of 12th ACM Workshop on Parallel and Distributed Simulation (PADS'98)*, pp. 154 – 161, 1998.
- [6] A. Boukerche and A. Fabbri. Partitioning parallel simulation of wireless network. In *Proc. of the 2000 Winter Simulation Conference*, pp. 1449 – 1457, 2000.
- [7] J. Panchal, O. Kelly, J. Lai, N. Mandayam, A. T. Ogielski, and R. Yates. WiPPET, a virtual testbed for parallel simulations of wireless networks. In *Proc. of the 12th workshop on Parallel and Distributed Simulation (PADS'98)*, pp. 162 – 169, 1998.
- [8] G. Karypis and V. Kumar. Multilevel k -way partitioning scheme for irregular graphs. in *the Journal of Parallel and Distributed Computing*, 1998.
- [9] METIS. <http://www-users.cs.umn.edu/~karypis/metis/index.html>.