論　文

MMORPGにおける動的領域分割結合アルゴリズム

榎原　博之[†]　　吉岡　　啓[†]　　松崎　頼人[†]

Dynamic Region Division and Binding Algorithm for MMORPG

Hiroyuki EBARA[†], Hiraku YOSHIOKA[†], and Raito MATSUZAKI[†]

あらまし　本論文では，P2P を用いた MMORPG (大規模仮想空間ロールプレイングゲーム) の動的な領域分割結合アルゴリズムについて提案する．P2P 型 MMORPG は，ゲームが行われる仮想空間を部分領域に分割し，ノードを管理ノードとして各部分領域に設置し管理させることでゲームを進行する．しかし，ノードの処理能力には限界があり処理能力を超える負荷がかかった場合，遅延の発生やゲームの中断につながるため，部分領域を再分割し負荷を分散する必要がある．そこで，各部分領域に部分領域内に存在できるプレイヤ数の上限と下限の 2 種類のしきい値を設けることでプレイヤの移動を検知し，動的に領域を分割・結合する負荷分散アルゴリズムを提案する．動的に領域分割することで，P2P 型通信の問題点であった負荷の集中に対応した負荷分散を行うことができる．シミュレーション実験による検証を行い，既知のアルゴリズムよりも提案アルゴリズムのほうが総負荷を軽減できることを示す．

キーワード　MMORPG，負荷分散，P2P 型通信，動的アルゴリズム

1. まえがき

近年，一般家庭への常時接続回線の急速な普及によって，大人数がネットワークを介して同時に大規模仮想空間内で遊ぶ多人数参加型オンラインゲーム (MMOG: Massively Multiplayer Online Games) が注目されている．1970 年代後半には CAI (Computer Aided Instruction) システムなどの応用によって，MMOG の基礎となる複数人が同じ仮想空間内でプレイできるゲームがアメリカで開発されている．1990 年代中盤以降の爆発的なインターネットの普及に伴い MMOG は発展し，数人から数百人で遊ぶシューティングゲームやパズルゲーム，ロールプレイングゲームなどの様々なソフトウェアが登場している．その中でも，オンライン RPG（MMORPG: Massively Multiplayer Online Roll Playing Game）は圧倒的な人気を誇り，最も加入者数の多い MMORPG である World of Warcraft [1] は，一時は全世界加入者数が 1200 万人にも及んでいる．

MMOG は通信接続方式によって「C/S (クライアント・サーバ) 型」と「P2P (Peer to Peer) 型」の 2 種類に分類することができる．現在の MMOG では一般的に C/S 型の通信接続方式が用いられている．この方式は，中央管理サーバにてゲームサービスを提供し，クライアントとなるプレイヤがサーバに接続することでゲームを進行する．中央管理サーバがセキュリティの確保やデータの管理を一括して行うので，コンテンツの管理が容易にできるという利点がある．しかし，MMORPG は他のジャンルと比較して大規模仮想空間に多くのプレイヤが同時に存在するため，スケーラビリティや探索クエリの増加によって帯域が圧迫されるといった問題が発生しやすい．そのため，全ての負荷がサーバに集中する C/S 型では，サーバの処理能力を超える負荷が発生し最悪の場合サービスが一時停止してしまうといった問題も発生しやすくなる．この問題は処理能力の高いサーバや広帯域ネットワークを確保することで補うことができるが，初期費用やネットワーク回線の維持費などが必要となりコストがかさむ．

そこで，C/S 型の通信方式にかわって，スケーラビリティの問題や帯域の圧迫が発生しにくい通信方式である P2P 通信接続方式を MMOG に適用した HybridP2P 型 MMORPG が検討されている．HybridP2P 型 MMORPG では，新規参入者の処理と各

[†] 関西大学，吹田市
Kansai University, 3–3–35 Yamate-cho, Suita-shi, 564–8680 Japan

プレイヤの状態情報や位置情報の保存を行うロビー・サーバと，プレイヤの中から選ばれたノード (以下，管理ノード) がゲームを進行する．管理ノードは，大規模なゲーム領域を複数に分割した領域 (以下，部分領域) ごとに設けられ，部分領域内で局所的な C/S 型を構築してゲームを進行する．ゲームの進行はロビー・サーバを介さずにプレイヤ間で負荷を分散するため，上記の C/S 型特有の問題を解決できる．しかし，管理ノードの処理能力はサーバに比べてはるかに低く，プレイヤの局所的な集中によって管理ノードの処理能力を超る負荷が発生し，ゲームが中断してしまうといった問題がある．また，プレイヤ接続切れ時のデータ保護，プレイヤ間で通信する際のデータの整合性の維持といった問題も発生する．このような問題を解消するため，今日までに幾つかの領域分割手法 [2]〜[5] や領域管理手法 [6]〜[12]，MMORPG 内のイベントを伝える際の通信負荷が一定になるよう通信のツリーを作成する分散型イベント配信手法 [13] などが検討，提案されている．

領域分割手法は領域を分割した後，分割された領域ごとに C/S 型を構築しゲームを進行する．領域管理手法はプレイヤひとりひとりが自身を管理し，ゲームを進行する．ゲームを進行する方式が異なるので，通信経路も異なってくる．領域管理手法では，個々の一般プレイヤが通信を行えるので，メッセージなどの個別の通信を管理ノードに中継して通信を行う領域分割手法に比べ負荷を低減することができる．しかし，ロビー・サーバ発の情報を送る際，管理手法は全てのプレイヤに情報を送る必要があり，ロビー・サーバの負荷が増加する．その点，領域分割手法では，管理ノードに情報を送るだけでよいので，ロビー・サーバの負荷を軽減できる．C/S 型の問題点であるサーバの負荷集中を解決するためには，Hybrid 型 P2P の処理手法においてサーバの負荷をより低減できる領域分割手法が望ましいと考えられる．

本論文では．P2P 型 MMORPG の問題点の一つである領域分割手法について，動的アルゴリズムを提案する．領域分割手法には,「四分木アルゴリズム [2]」(以下，四分木) や,「JoHNUM [3]」,「Microcell [4]」が既に提案されているが，本論文では動的に領域を分割・結合するアルゴリズムを提案する．提案アルゴリズムは，部分領域内に存在できるプレイヤ数の上限と下限の 2 種類のしきい値を設けることでプレイヤの移動を考慮した動的な領域分割結合を行う．提案アルゴリズムを用いることで既知の手法よりも，管理ノードを減らし，総負荷の削減と各管理ノードの負荷分散の低減を図る．

本論文は以下の **6.** で構成される．**2.** では，関連研究を述べる．**3.** では，2 種類の管理ノードの役割などについて述べた後，提案アルゴリズムについて述べる．**4.** では，提案アルゴリズムの有望性を述べる．**5.** では，シミュレーションによる性能評価及び考察を行う．最後に **6.** では，本論文のまとめと今後の課題について述べる．

2. 関 連 研 究

MMORPG における負荷分散手法には，提案アルゴリズムと同様に領域に着目した P2P 型の「四分木 [2]」，C/S 型の「Microcell [4]」と「JoHNUM [3]」といった領域分割手法が存在する．

四分木とは，管理ノードに処理能力以上の負荷がかかると部分領域 (正方形) を四つの部分領域 (正方形) に分割する領域分割手法である．この作業を複数回繰り返すことによって，プレイヤの局所的な密集による負荷の発生にも対応することができる．しかし，四分木は部分領域の分割しか行わないため，部分領域数を減らすことができない．そのため，密集状態が解消された後も分割後の小さくなった部分領域がそのまま残り，プレイヤ数に対して必要以上の部分領域と管理ノードが残るという問題が生じる．管理ノードは自身が管理する部分領域内のプレイヤが他の管理ノードが管理する部分領域に移動する際，移動先の管理ノードにプレイヤデータの送信を行う．部分領域数が多いと管理ノード間のプレイヤデータの通信頻度が増え負荷が増加する問題が発生する．図 1 にプレイヤ密集発生時の分割とその後プレイヤが移動した例を示す．図 1 の左上の部分領域に注目すると，プレイヤの局所的な

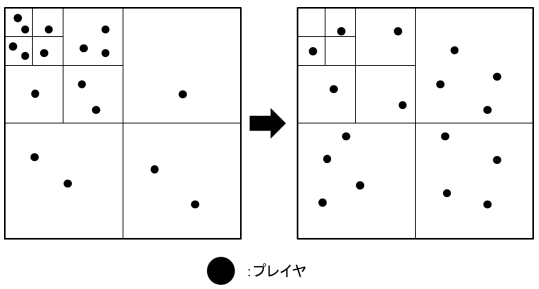

図 1 四 分 木
Fig. 1 Quadtree algorithm.

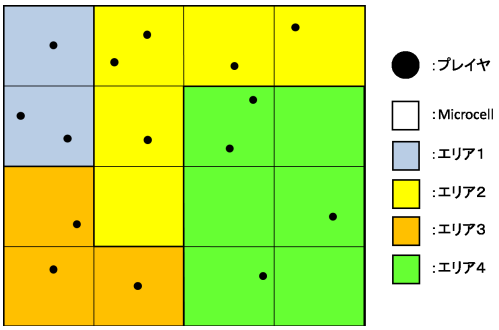

図 2　Microcell
Fig. 2　Microcell.

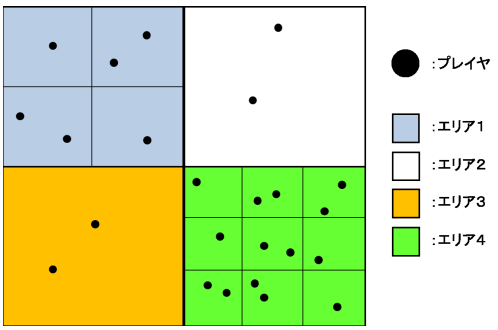

図 3　JoHNUM
Fig. 3　JoHNUM.

密集発生後，プレイヤが移動し負荷の集中が解消されても分割された部分領域が残ってしまっていることがわかる．

Microcell [4] は，事前に領域を一定の大きさの部分領域に分割しておき，プレイヤの密集に合わせて領域全体を見直し，部分領域の形を変化させるアルゴリズムである．しかし，このアルゴリズムは部分領域の形状を考慮していないため，プレイヤの位置によっては部分領域の形が歪になる．部分領域の形が歪だと，プレイヤが管理ノードが異なる部分領域に移動する頻度が高くなり，管理ノード間のプレイヤデータの通信負荷が増加する問題がある．図 2 に Microcell の分割例を示す．

JoHNUM は，事前に領域を四つのエリアに分割しておき，それぞれのエリア内の部分領域を別々に分割することで負荷の分散を図るアルゴリズムである．管理ノードに処理能力以上の負荷がかかると，その管理ノードが属するエリア内を，4，9，16，25 . . . というように現在のエリア内の部分領域数の平方根を 1 増加させ 2 乗した数の部分領域に分割する．四分木 [2] と同様に，部分領域の分割しか行わないためプレイヤの密集には対応できるが，プレイヤの過疎には対応できないという問題点がある．図 3 に JoHNUM の分割例を示す．

また，ゲーム領域以外の様々な要素に着目した分割手法も提案されている．Thawonmas 氏らはプレイヤには行動パターンが存在し，その行動パターンを用いた分割手法 [5] を提案している．領域を m × n 格子の部分領域に分割し，プレイヤの移動ログから頻繁にプレイヤの訪問があった部分領域を目印として設定すると，目印は特にゲームで重要な位置の周辺に現れる．また，目印間の重みから推移確率を計算し，推移確率 (動作パターン) に基づいてプレイヤクラスタを作成すると，ほとんどのプレイヤを数個のクラスタに分割することができる．以上より，ほとんどのプレイヤに同様の動作パターンがあることを示し，ゲーム内に目印を設定することによってプレイヤの群生を操作することで負荷の分散を図っている．

このプレイヤの動作パターンを用いて，AOI (Area of Interest) と呼ばれるデータの整合性を維持するための領域を設置，変形する方法が Dewan Tanvir Ahmed 氏らの研究 [6] によって提案されている．各プレイヤの移動ログから行動範囲が似ているプレイヤ群を一つのグループとする．あらかじめ任意の場所に AOI を設置しておき，グループの行動を基に AOI を変形させていくことで，動的に AOI を作成するというものである．同じ動作パターンのプレイヤは互いに近接する頻度が多いことから，同じ動作パターンのプレイヤをグループ化して，グループに対する管理ノードを設定しプレイヤを管理するグループベースの負荷分割処理手法 [7] も提案されている．また，上記で紹介した四分木や JoHNUM，グループベースの負荷分割処理手法では，特定のノードに負荷を処理させることでゲームを進行するが，特定の管理ノードを設定せずに各プレイヤが自分自身を管理しゲームを進行する方式 [8] も提案されている．

グループベースの負荷分割処理手法 [5]～[7] は，通信頻度の高いプレイヤ同士を同じグループに所属させることで通信負荷を軽減を図る．これらの手法は個々の一般プレイヤ同士が通信を行えるので，メッセージなどの個々の通信を管理ノードに中継して通信を行う領域分割手法に比べ負荷を低減することができる．しかし，プレイヤが広範囲に影響を及ぼすアクションを行う際，影響を及ぼす範囲内の全プレイヤと通信を行

わなければならないため，アクションを行うプレイヤの通信負荷が瞬時に増加する．各プレイヤは一般プレイヤであり，処理能力があまり高くないと予測されることからこれは大きな問題である．それに対して，領域分割手法は各管理ノードが分割してゲーム状態の通知を行うため，プレイヤが広範囲に影響を及ぼすアクションを行った場合でもプレイヤ個人にかかる負荷は小さい．また，各プレイヤが自分自身を管理しゲームを進行する方式 [8] も同様に，プレイヤが広範囲に影響を及ぼすアクションを行う際に通信負荷が瞬時に増加する問題がある．大人数が同時に大規模仮想空間内で遊ぶ MMORPG では，複数人に影響を及ぼすアクションやイベントが多くあると考えられることから，グループベースの負荷分割処理手法より提案アルゴリズムの方式である領域分割手法が適していると考えられる．

3. 領域分割手法

P2P 型の通信方式は，一般プレイヤから選ばれた管理ノードがゲーム内の負荷を分散して処理することでゲームを進行する．管理ノード数を削減できれば，管理ノード間の通信負荷が削減でき，結果的に総負荷の削減が可能である．しかし，管理ノードを減らすと 1 台の管理ノードの負荷が大きくなる．提案アルゴリズムは，1 台の管理ノードの負荷が従来法を超えることを容認しつつ，負荷のばらつきを押さえるとともに，管理ノード数を削減することにより総負荷を従来法より下げることを目的とする．

3.1 前　　提

本論文では，プレイヤのログイン/ログアウトを管理し，現在ゲームを連結する全てのノードのリストを維持するロビー・サーバが存在する HybridP2P 型 MMORPG を採用する．また，ゲーム進行は多数のプレイヤが同じゲーム領域及び時間を共有するタイムスロット方式を用いる．全てのプレイヤはタイムスロットの時間間隔により通信を行い，自身の状態情報の更新や行動の通知を行う．本論文における P2P 型 MMORPG での領域分割手法では，大規模なゲーム領域を複数の部分領域に分割してゲームを進行するため，部分領域の分割・結合処理を行うノードと分割した領域を管理するノードを設定する．処理を行うノードは一般プレイヤの中で計算能力が高く，通信帯域に余裕のあるノードが選定されることが望ましい．そこで，ロビー・サーバが維持するリストには，各ノードの計算能力，通信帯域，滞在時間なども保持されているものとし，それらの値を参考に処理を行うノードを決定する．また，仮想空間は 2 次平面であり，横軸 (x 軸) と縦軸 (y 軸) からなるものとし，提案アルゴリズムはマス目単位で領域を扱うものとする．プレイヤは各マス目に 1 人しか存在できないものとする．

3.1.1 分割管理ノードと領域管理ノード

部分領域の分割・結合とそれに関連する処理を行うノードを分割管理ノード，領域を分割してできた部分領域を管理するノードを領域管理ノードと呼ぶ．領域管理ノードは部分領域ごとに設けられ，その部分領域内でサーバの役割を果たす．分割管理ノードは一般プレイヤからロビー・サーバによって 1 台選定され，選定時にロビー・サーバから領域管理ノードのリストを得るものとする．また，領域管理ノードは一般プレイヤから分割管理ノードにより選定され，選定時に自身が管理する部分領域内のプレイヤのリストを作成するものとする．分割管理ノード，領域管理ノードは一般のプレイヤと同じように移動などのアクションを行う．したがって，領域管理ノードは自身が管理している部分領域内に滞在している必要はなく，別の部分領域に属する場合もある．領域管理ノードの役割を以下に示し，分割管理ノード，領域管理ノード，プレイヤ間の各通信のタイミングと通信内容を表 1 に示す．相互干渉領域については次節で説明する．

(1) 自身が管理する部分領域内のプレイヤ数が大きくなると，分割管理ノードに分割要求を送信する．

(2) 部分領域分割後，各領域管理ノードは自身が管理する部分領域内のオブジェクトデータ (以下，領域データ) をその部分領域に属する全プレイヤに送信する．

(3) 部分領域内に存在するプレイヤと隣接する部分領域の領域管理ノードの IP アドレスと領域範囲を取得しリストを作成する．

(4) 移動によってプレイヤが相互干渉領域に属した場合，プレイヤに相互干渉領域に対応する領域管理ノードの IP アドレスを送信する．

3.1.2 相互干渉領域

P2P 型では，大規模仮想空間を部分領域に分割し，各部分領域に管理ノードを割り当て管理することでプレイヤが仮想空間内を自由に移動できるようにしている．しかし，ただ部分領域を分割するだけでは，プレイヤが異なる領域管理ノードが管理している部分領域へ移動する際にリアルタイムな情報更新ができず，

表 1　通信タイミングと通信内容
Table 1　The timing and content of communication.

タイミング	送信ノード ⇒ 受信ノード	通信内容
毎タイムスロット	プレイヤ ⇒ 領域管理ノード	プレイヤの行動情報，状態情報，位置情報
	領域管理ノード ⇒ プレイヤ	ゲーム状態
領域分割時	分割管理ノード ⇒ 領域管理ノード	管理を割り当てた部分領域情報と相互干渉領域情報
	領域管理ノード ⇒ プレイヤ	領域管理ノードの IP アドレス
相互干渉領域内にプレイヤ移動時	領域管理ノード ⇒ プレイヤ	相互干渉領域に対応する領域管理ノードの IP アドレス
	プレイヤ ⇒ 相互干渉領域に対応する領域管理ノード	プレイヤの行動情報，状態情報，位置情報

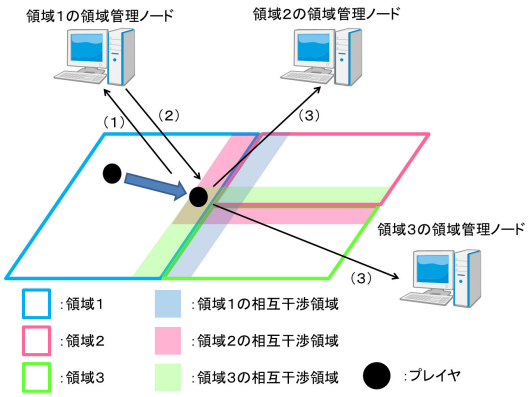

図 4　相互干渉領域
Fig. 4　Mutual interference area.

データに誤りが生じる場合がある．そこで，データの整合性を維持するために Dewan Tanvir Ahmed 氏らの研究 [6] で導入された相互干渉領域を本論文にも導入する．相互干渉領域はプレイヤの移動などのアクションによりプレイヤデータの通信が予想される場合，事前にプレイヤデータを通信しておくことでデータの整合性を維持する領域である．ただし，Dewan Tanvir Ahmed 氏らが導入した相互干渉領域はグループベースの負荷分割処理手法に適応するよう各プレイヤごとに設定されていたが，領域分割手法である提案アルゴリズムには適応しない．そこで，提案アルゴリズムに適応するように，相互干渉領域は各部分領域ごとに設定されるよう仕様を変更して導入する．図 4 にプレイヤデータの受け渡し順序と流れを示す．

(1)　相互干渉領域内にプレイヤが移動した際に，プレイヤは自身が属している部分領域の領域管理ノードにプレイヤの移動を連絡する．

(2)　プレイヤから移動連絡を受けとった領域管理ノードは隣接している領域管理ノードの IP アドレスをプレイヤに送信する．

(3)　相互干渉領域内にいるプレイヤは，IP アドレスを基に隣接している領域管理ノードに自身のプレイヤデータを送信する．以後，相互干渉領域内にいる限り各プレイヤは自身が属する部分領域と，隣接している領域管理ノードと通信を行う．

この相互干渉領域を設けることで，領域管理ノード間で整合性のとれた情報更新が可能となる．

3.2　動的領域分割結合アルゴリズム (提案アルゴリズム)

部分領域の分割・結合は部分領域が内包するプレイヤ数を考慮して動的に行う必要がある．そこで，分割しきい値，結合しきい値，領域プレイヤ数の三つの値を設定し，部分領域内のプレイヤ数に対応して部分領域数を増減する動的な分割結合アルゴリズムを提案する．ゲーム領域全体に対して部分領域の分割・結合を行うと一度に高負荷がかかる．そこで，しきい値を超えた部分領域とその部分領域に隣接する部分領域を分割結合対象領域として指定し，分割・結合を行う際はこの分割結合対象領域に対して分割を行う．複数の部分領域で分割・結合処理が開始され分割結合対象領域がバッティングした場合，バッティングした分割結合対象領域を結合させ分割･結合を行う．また，動的領域分割結合アルゴリズムの計算中は既存の領域管理ノードが部分領域と相互干渉領域を引き続き管理し，分割結果が求まり対象の領域管理ノードと同期がとれ次第ゲームに反映されるものとする．以下に，分割しきい値，結合しきい値，領域プレイヤ数について説明する．

- 分割しきい値

任意に決めた各部分領域内に存在できるプレイヤ数の上限値を示す．

- 結合しきい値

任意に決めた各部分領域内に存在できるプレイヤ数の下限値を示す．

- 領域プレイヤ数

部分領域を分割する際に各部分領域が内包する標準プレイヤ数を示す．提案アルゴリズムではこの値を上限として領域を分割する．分割しきい値と結合しきい値の平均の切り上げ値とする．

また，分割しきい値 ≥ 2 × 結合しきい値 と仮定する．

以下に本論文で提案するアルゴリズムを示す．

(1) プレイヤの移動 (または新規参入・離脱) によって，部分領域内のプレイヤ数が分割しきい値を超える，または結合しきい値を下回る．

(2) しきい値を超えた部分領域の領域管理ノードは，分割管理ノードに分割要求メッセージを送信する．

(3) 分割管理ノードは，しきい値を超えた部分領域とその部分領域に隣接する部分領域の領域データを取得する．

(4) プレイヤの座標データなどを受け取った分割管理ノードは，自身が管理する部分領域を中心に分割結合対象領域を作成する．

(5) 分割結合対象領域に対して後述する分割開始マス決定を行い，分割を開始するマス (以下，分割開始マス) と分割を進める方向 (以下，分割方向) を決める．

(6) 分割開始マス決定で決定した分割開始マスと分割方向を基に，後述する領域分割を行う．

(7) 後述する領域結合を行う．

(8) 分割管理ノードは自身が所有するリストを基に新しくできる部分領域に領域管理ノードを任命し，自身のリストを更新する．

(9) 新しく任命された領域管理ノードは，自身が管理する部分領域に属するプレイヤのリストを作成する．

(10) 分割管理ノードは新しく任命された領域管理ノード及び既存の領域管理ノードと通信し，分割結果と相互干渉領域をゲーム内に反映する．

(11) 分割結果反映後，各領域管理ノードは自身が管理する部分領域に属するプレイヤに IP アドレスと新たな領域データを送信する．以後，各プレイヤは IP アドレスが送られてきた領域管理ノードと通信を行う．

(12) プレイヤに領域データ送信後，プレイヤの移動等により，各部分領域内のプレイヤ数が分割しきい値を超える，または結合しきい値を下回った場合，(1)〜(11) の操作を繰り返し行う．

以下に図 5 の (a)〜(g) を用いて動的領域分割結合アルゴリズムを例で説明する．以降，領域分割手法やアルゴリズムを説明する際に用いる図では，分割しきい値 5，結合しきい値 1，領域プレイヤ数 3 と仮定する．

(1) プレイヤが領域 5 に移動する (図 5 (a))．

(2) 領域 5 内のプレイヤ数が分割しきい値を超える (図 5 (b))．

(3) 領域 5 の領域管理ノードは，分割管理ノードに分割メッセージを送信する．

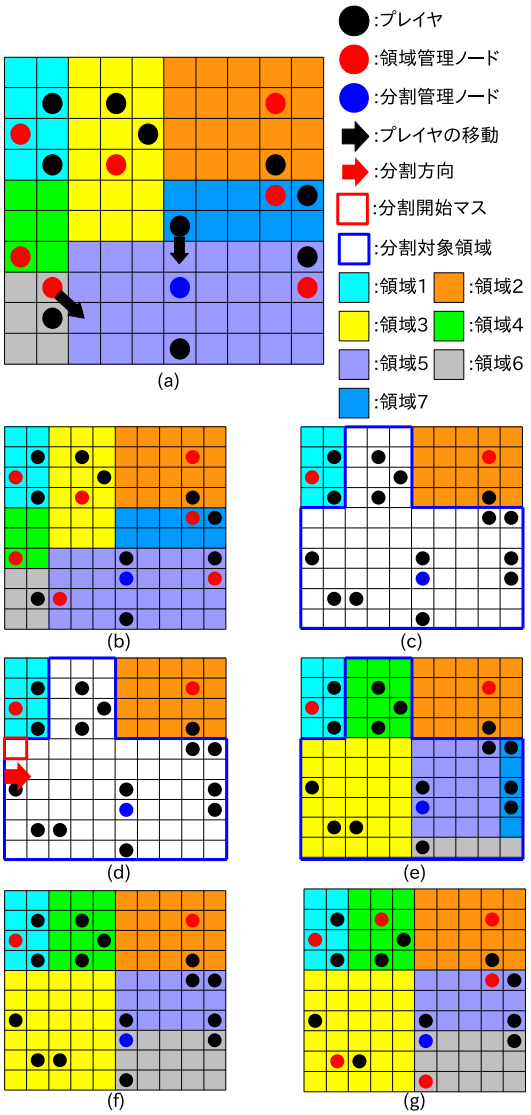

図 5 動的領域分割結合アルゴリズム
Fig. 5 Extended dynamic region decomposition algorithm.

(4) 分割管理ノードは分割しきい値をこえた領域 5 と，その隣接領域 3，4，6，7 を分割結合対象領域として指定する (図 5 (c))．

(5) 分割結合対象領域に対して，後述する分割開始マス決定を行う (図 5 (d))．

(6) 分割開始マス決定の結果を基に後述する領域分割を行う (図 5 (e))．

(7) 分割結合対象領域内の部分領域に対して，後述する領域結合を行う (図 5 (f))．

(8)　分割管理ノードは自身が所有するリストを基に新しくできる部分領域に領域管理ノードを任命し，自身のリストを更新する (図 5 (g))．

(9)　新しく任命された領域管理ノードは，自身が管理する部分領域に属するプレイヤのリストを作成する．

(10)　分割管理ノードは新しく任命された領域管理ノード及び既存の領域管理ノードと通信し，分割結果と相互干渉領域をゲーム内に反映する．

(11)　分割結果反映後，各領域管理ノードは自身が管理する部分領域に属するプレイヤに IP アドレスと新たな部分領域と相互干渉領域の領域データを送信する．以後，各プレイヤは IP アドレスが送られてきた領域管理ノードと通信を行う．

3.2.1　分割開始マス決定

分割結合対象領域の形によっては，領域分割を行うと部分領域数が多くなりやすく，細長い部分領域ができやすいものがある．細長い部分領域はプレイヤが移動する際に異なる領域管理ノードが管理する部分領域を跨ぐ頻度を増加させ，領域管理ノード間の通信負荷の増加を引き起こすという問題点がある．そこで，分割結合対象領域の形によって分割開始マスと分割する際の部分領域を広げていく順番を変更することで，ある程度細長い部分領域をできにくくすることができる．なお，x 軸方向，y 軸方向の順に部分領域を広げ，分割を進めていくことを横方向とし，y 軸方向，x 軸方向の順に部分領域を広げ，分割を進めていくことを縦方向とする．分割開始マスと分割方向の違いによる領域分割後の部分領域の違いを図 6 で示す．領域分割については後で述べる．

図 6 より，分割の開始マスと分割方向によって同じ形の分割結合対象領域でも，領域分割後の部分領域数や部分領域の形が大きく異なることがわかる．また，それぞれの領域分割後を見ると「最も左の上から横方向に領域分割」の方が正方形に近い部分領域数が多く，「最も上の左から縦方向に領域分割」したものより望ましい分割結果であるといえる．分割結合対象領域に他の部分領域によって凹まされている部分 (以下，凹部分) がある場合，凹部分の y 辺に沿って部分領域を作成すると，凹部分の x 辺と同様の長さをもつ部分領域ができやすくなる．凹部分の x 辺に沿って部分領域を作成すると，凹部分の y 辺と同様の長さをもつ部分領域ができやすくなる．その理由を図 7 を用いて説明する．

図 7 の領域 1 の y 辺に沿って部分領域を作成する

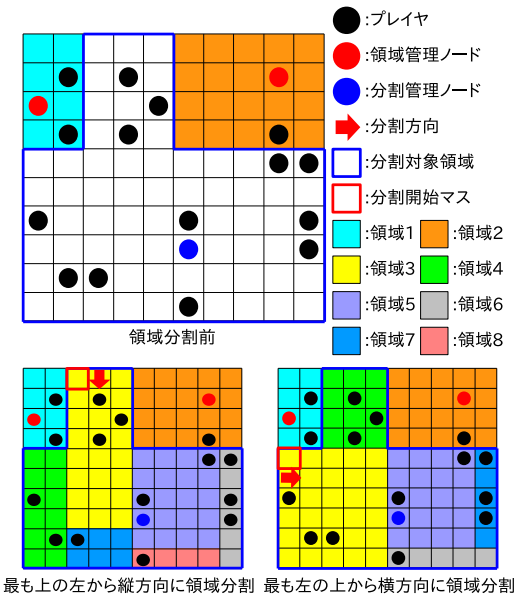

図 6　分割開始マスと分割方向
Fig. 6　Starting place and dividing direction.

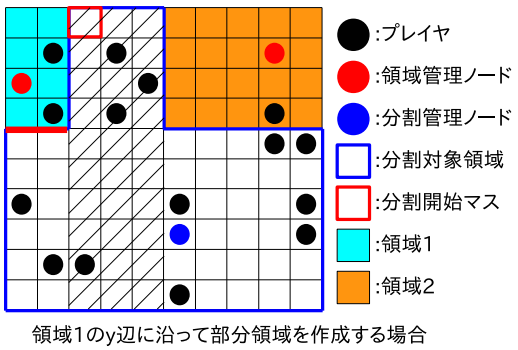

図 7　部分領域作成の性質
Fig. 7　The nature of the area divided.

場合を考える．領域 1 の y 辺に沿って部分領域を広げていくと，図 7 の黒斜線の範囲に部分領域が作成される．分割結合対象領域の左部分に部分領域を作成する際に，この黒斜線の部分領域が新たな部分領域の横方向の広がりを止めるため，凹部分の x 辺 (赤色線) を一辺とする部分領域が作成されやすくなる．凹部分の短辺に沿って分割を進めるように分割開始マスを決定することで，細長い部分領域の作成をある程度防ぐことができる．

図 8 の (a)～(c) を用いて分割開始マス決定のアルゴリズムを説明する．

(1)　分割結合対象領域の左上，右上，左下，右下の

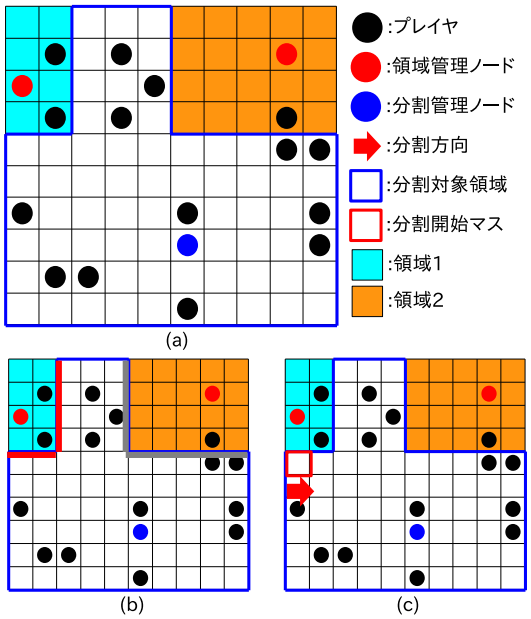

図 8　分割開始マス決定のアルゴリズム
Fig. 8　Algorithm of divided decision.

それぞれに凹部分があるか調べる (図 8 (a)).

(2)　左上 (図 8 (b) の赤色線) と右上 (図 8 (b) の灰色線) に凹部分があるため，それぞれの凹部分の x 長と y 長の差の絶対値を求める (図 8 (b)).

(3)　(2) で求めた絶対値が最も大きい凹部分の短辺の外側のマスを分割開始マスとし，短辺に沿った方向を分割方向とする (図 8 (c)).

分割方向は実質的には重要ではなく，分割方向による性能差はほとんどない．提案アルゴリズムでは，上記のように短辺に沿った方向を分割方向としている．その理由は，辺と接するまでの距離が長いからである．

3. 2. 2　領 域 分 割

分割開始マス決定後，分割結合対象領域に対して領域分割を行う．領域プレイヤ数を超えない範囲で x 長，y 長を交互に増加させる．また，部分領域の x 長を増加中に他の部分領域に接した場合，x 長の増加を停止し y 長のみを増加させ部分領域の範囲を広げる．同様に，部分領域の y 長を増加中に他の部分領域に接した場合，y 長の増加を停止し x 長のみを増加させ部分領域の範囲を広げる．このような条件を加えることで，部分領域を必ず四角形に保つことができる．以下に図 9 の (a)～(g) を用いて領域分割のアルゴリズムを説明する．

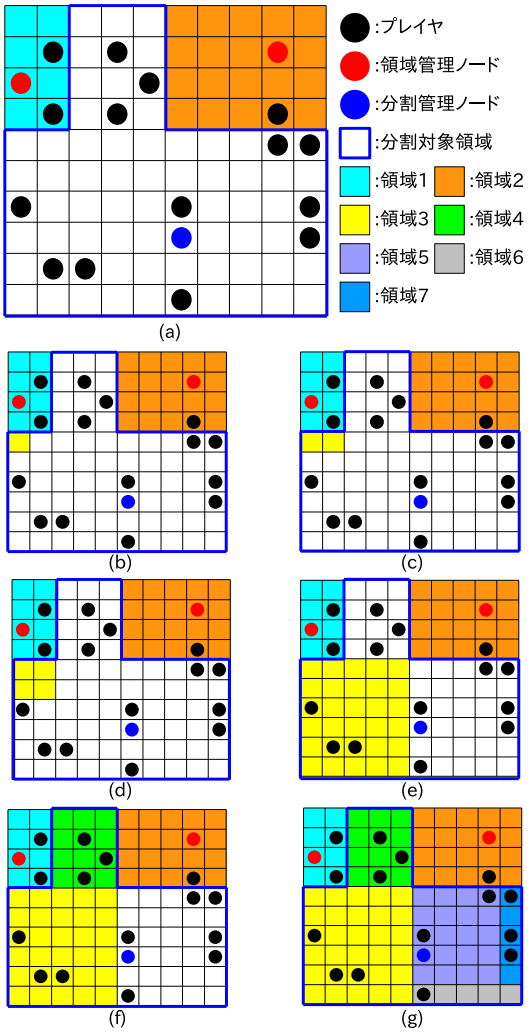

図 9　領域分割のアルゴリズム
Fig. 9　Extend dividing algorithm.

(1)　分割結合対象領域の有無を調べ，分割開始マスと分割方向を決定する (図 9 (a)).

(2)　分割開始マス (図 9 (b) の黄マス) から分割方向 (横方向) に分割を開始する．

(3)　部分領域である領域 3 の x 長を 1 マス増加させる (図 9 (c)).

(4)　領域 3 の y 長を 1 マス増加させる (図 9 (d)).

(5)　領域 3 が内包するプレイヤ数が領域プレイヤ数を超えない，または他の部分領域と接さない限り (3), (4) の動作を繰り返す．

(6)　領域プレイヤ数に達した領域 3 は，x 長も y 長もプレイヤか辺に接するまで領域を広げる (図 9 (e)).

(7) 残った分割結合対象領域に対して，分割方向が左からの横方向の場合は最も左で上のマスを分割開始マスとして縦方向で，(3)〜(6) と同様に次の部分領域 (領域 4) の作成を行う (図 9 (f))．分割方向が上からの縦方向の場合は最も上で左のマスを分割開始マスとして横方向で，(3)〜(6) と同様に次の部分領域の作成を行う．分割方向が右からの横方向の場合は最も右で下のマスを分割開始マスとして縦方向で，(3)〜(6) と同様に次の部分領域の作成を行う．分割方向が下からの縦方向の場合は最も下で右のマスを分割開始マスとして横方向で，(3)〜(6) と同様に次の部分領域の作成を行う．

(8) 分割結合対象領域が残っている間は，残っている領域について (7) の方法で分割開始マスと分割方向を決定し，(2)〜(7) を繰り返す (図 9 (g))．

3.2.3 領域結合

領域分割は内包するプレイヤ数に加えて部分領域の形も考慮するため，場合によっては内包するプレイヤ数が結合しきい値を下回る部分領域や，細長い部分領域ができてしまう．そこで，分割結合対象領域内の結合しきい値を下回る部分領域と，部分領域の x 長と y 長の比が 1:2 より小さい，または 2:1 より大きい部分領域を隣接する部分領域と結合する領域結合を行う．領域が四角形でない場合は，その領域を囲む四角形の x 長，y 長とする．

提案する領域結合アルゴリズムは，ベストエフォート型のアルゴリズムである．しきい値を満たすことを優先し，領域の縦横比が 2 を超えることを許容する．更に，領域が四角形にならない場合も存在する．

以下に，領域結合の流れを示す．

(1) 分割結合対象領域内の結合しきい値を下回る部分領域，あるいは部分領域の x 長と y 長の比が 1:2 より小さい，または 2:1 より大きい部分領域を検知し，検知された部分領域の領域番号の順に以下の領域結合を行う．

(2) 検知した部分領域が縦長（$x < y$）の場合は，結合後の部分領域の頂点数が最も少なくなる左右どちらかの辺に接する全ての部分領域とそれぞれ結合する．結合後にできる部分領域の頂点数が同じ場合は左の部分領域と結合する．

検知した部分領域が横長（$x > y$）の場合は，結合後の部分領域の頂点数が最も少なくなる上下どちらかの辺に接する全ての部分領域とそれぞれ結合する．結合後にできる部分領域の頂点数が同じ場合は上の部分領域と結合する．

結合する部分領域が正方形（$x = y$）の場合は，結合後の部分領域の頂点数が最も少なくなる上下左右どちらかの辺に接する全ての部分領域とそれぞれ結合する．結合後にできる部分領域の頂点数が同じ場合は左上右下の優先順位で部分領域と結合する．

(3) 結合後の部分領域が内包するプレイヤ数が分割しきい値より多い場合，その部分領域の形が縦長か横長かを調べる．

(4) 横長（$x > y$）なら部分領域を縦に二等分する．同様に，縦長または正方形（$x \leq y$）なら部分領域を横に二等分する．

(5) 二等分後の部分領域の内包するプレイヤ数が分割しきい値より多い場合は，分割前の部分領域が横長（$x > y$）なら 1 列ずつ，縦長または正方形（$x \leq y$）なら 1 行ずつプレイヤ数の少ない部分領域からプレイヤ数の多い部分領域に，内包するプレイヤ数が分割しきい値以下になるまで部分領域の範囲を広げていく．

(6) (5) の処理後，内包するプレイヤ数が分割しきい値以下にならない場合，部分領域を二等分する前の状態に戻し，領域を縦長または正方形（$x \leq y$）なら縦に，横長（$x > y$）なら横に二等分する．

(7) 二等分後の部分領域の内包するプレイヤ数が分割しきい値より多い場合は，分割前の部分領域が横長（$x > y$）なら 1 行ずつ，縦長または正方形（$x \leq y$）なら 1 列ずつプレイヤ数の少ない部分領域からプレイヤ数の多い部分領域に，内包するプレイヤ数が分割しきい値以下になるまで部分領域の範囲を広げていく．

(8) (1) の条件で検知した全ての部分領域に対して (2)〜(7) を行う．

図 10 の (a)〜(d) を用いて領域結合のアルゴリズムを説明する．

(1) 領域 6，7 が部分領域の x 長と y 長の比が 1:2 より小さい，または 2:1 より大きくなるので検知する (図 10 (a))．

(2) 検知した領域の結合する方向を決める．図 10 では，領域 6 が横長なので結合する方向を縦方向とし，領域 7 が縦長なので結合する方向を横方向とする．

(3) 領域 6 を上の部分領域（領域 5 と領域 7）と結合する (図 10 (b))．

(4) 領域 7 を左の部分領域（領域 5）と結合する (図 10 (c))．

(5) 部分領域の結合によって領域 5 が内包するプレイヤ数が分割しきい値を超える，領域 5 は縦長なので

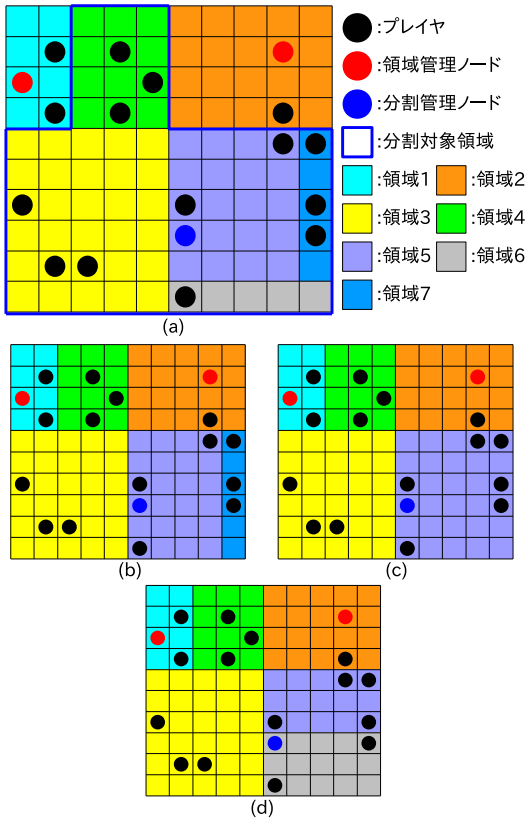

図 10　領域結合のアルゴリズム
Fig. 10　Algorithm of regious reformation.

横に分割する (図 10 (d)).

この領域結合アルゴリズムは，しきい値を満たすことを保証していない．その理由は，提案アルゴリズムは，1 行ごと（1 列ごと）に領域を広げていくため，プレイヤがある行（ある列）にしきい値の範囲を超える数並んでいた場合，しきい値を満たすことが不可能となるからである．その場合，提案アルゴリズムでは，次のタイムスロットで処理することになる．次のタイムスロットでは，プレイヤも移動しているため，同様の問題は解消される．

4. 部分領域の分割・結合による通信回数の増減

提案アルゴリズムと **2.** で示した従来法である「四分木 [2]」や「JoHNUM [3]」が大きく異なる点はプレイヤの密集に対応して部分領域を分割し負荷を分散させるだけでなく，プレイヤの過疎に対応して部分領域を結合する点である．本章では，部分領域の分割・結合によって **3. 1. 1** の表 1 で記述したゲームを進行する上で生じる通信の回数にどのような増減が見込まれるか記述し，提案アルゴリズムの有望性について検討する．

まず，**3. 1. 1** の表 1 で記述した各ノード間の通信について説明する．

- 領域管理ノード–プレイヤ間の通信

領域管理ノードは自身が管理する部分領域内の各プレイヤ情報を取得し，各プレイヤに現在のゲーム状態を通知するため一定の時間間隔でプレイヤと通信を行う．この通信回数は部分領域が内包するプレイヤ数にのみ依存する．

- 分割管理ノード–領域管理ノード間の通信

分割管理ノードが新しく部分領域を作成し領域管理ノードに管理を割り当てる際に，作成した部分領域情報と相互干渉領域情報の通信を行う．

- 相互干渉領域に対応する領域管理ノード–プレイヤ間の通信

相互干渉領域内にプレイヤが移動してきた場合に相互干渉領域に対応する領域管理ノードは自身が管理する部分領域内へのプレイヤの移動に備え，そのプレイヤデータを取得するため通信を行う．

次に，部分領域の分割・結合によって部分領域数，各部分領域が内包するプレイヤ数，相互干渉領域が内包するプレイヤ数がどのように増減するか図 11 に示す．なお，図 11 は部分領域の分割・結合によるゲーム全体の通信回数の増減に着目するため，各相互干渉領域を混合して表示している．しかし，実際の相互干渉領域は「領域管理ノード A が管理する部分領域の相互干渉領域」,「領域管理ノード B が管理する部分領域の相互干渉領域」といったように別々のものである．

図 11 から，部分領域の分割を行うことで部分領域数が増加し，各部分領域が内包するプレイヤ数の平均が減少，相互干渉領域が内包するプレイヤ数は増加することがわかる．よって，部分領域の分割を行うことで領域管理ノード–プレイヤ間の通信回数の減少と相互干渉領域に対応する領域管理ノード–プレイヤ間の通信回数の増加が見込まれる．また，部分領域の結合を行うことで部分領域数が減少し，各部分領域が内包するプレイヤ数の平均が増加，相互干渉領域が内包するプレイヤ数は減少することがわかる．よって，部分領域の結合を行うことで領域管理ノード–プレイヤ間の通信回数の増加と相互干渉領域に対応する領域管理ノード–プレイヤ間の通信回数が減少が見込まれる．

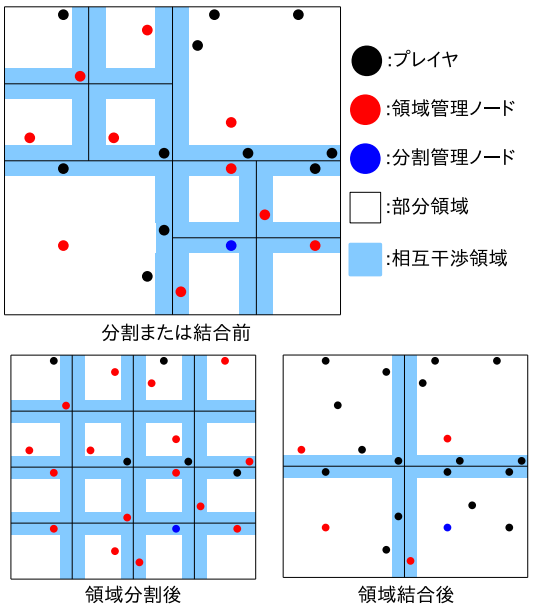

図 11　通信対象となるプレイヤ数の増減
Fig. 11　Increase or decrease in the number of players to be communicated.

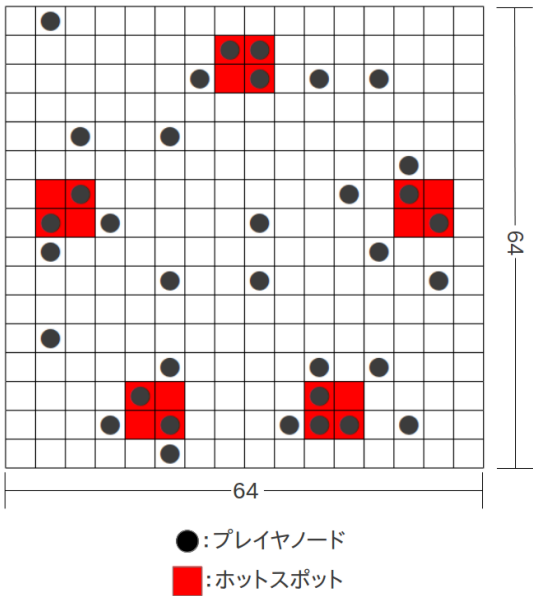

図 12　設定したホットスポットの位置
Fig. 12　Hotspot.

従来法はプレイヤ数が減少した場合に，それまでに増加した部分領域数を減少させることができないので，余計な部分領域が残ってしまう．したがって，プレイヤ数に対して相互干渉領域に対応する領域管理ノード–プレイヤ間の通信回数が多くなる．提案アルゴリズムは部分領域の分割・結合の両方を行うので，従来法に比べて分割管理ノード–領域管理ノード間の通信回数が多くなるが，それ以上に相互干渉領域に対応する領域管理ノード–プレイヤ間の通信回数が減少できると考えられる．

5. 性能評価

提案アルゴリズムの有効性を示すため，コンピュータシミュレーションによる性能評価を行う．シミュレーションでは，**2.** で説明した四分木と JoHNUM による領域分割手法，提案提案アルゴリズムである動的領域分割結合アルゴリズムの三つをそれぞれ比較検討し，それらの性能について評価する．

5.1　シミュレーションモデル

シミュレーションに際し，仮想空間，タイムスロット，プレイヤの移動，相互干渉領域，負荷の設定を以下に示す．

(1)　仮想空間

仮想空間は 64×64 マスの正方形の 2 次元平面とし，領域の最小単位を 1 マスとする．また，現実世界で人が不快なく存在するためには，ある程度空間が必要である．その快適な空間のことをパーソナル・スペース [14] とよび，一般的に 45〜75cm とされている．よって仮想空間内の 1 マスを 60cm とする．

(2)　タイムスロット

シミュレーション実験において，各プレイヤの移動，通信の時間間隔を指す．本実験では，0.5 秒に設定する．

(3)　プレイヤの移動パターンと移動速度

プレイヤの移動パターンは，以下の 2 パターンである．

－　通常移動

MMORPG ではグループ単位で行動することが多く，プレイヤは一様に分布しにくい．そこで，プレイヤの振る舞いを完全なランダムにせず，動作パターンにある程度の特徴をもたせるため，図 12 のようなホットスポットを 5ヶ所設定する．ホットスポットとは，プレイヤノードが新規参入しやすい場所，ゲーム上での重要な拠点などを指し，プレイヤが集中するエリアである．プレイヤの移動は，確率に従い上下左右，斜め移動または移動しないの 9 パターンの行動をランダムに選択する．各々の行動の確率は 1/10 であり，残りの 1/10 の確率でプレイヤにあらかじめ決められた

ホットスポットへ向かう方向の移動を選択する．また，プレイヤに決められたホットスポットは 1/10 の確率で別のホットスポットに変わるものとする．

– ジグザグ移動

相互干渉領域の境目を頻繁に移動した場合の領域管理ノードのオーバーヘッドを調べるため，プレイヤがジグザグに移動する場合を考える．x 軸方向にジグザグになるように斜め移動し，y 軸方向にジグザグになるように斜め移動することでホットスポットに向かうパターン 1 とし，y 軸方向にジグザグになるように斜め移動し，x 軸方向にジグザグになるように斜め移動することでホットスポットに向かうパターン 2 とする．各プレイヤはあらかじめ決められた 1，2 のどちらかのパターンで移動する．

(4) プレイヤの移動速度と相互干渉領域

– 移動速度

全てのプレイヤの移動速度は一定とする．プレイヤの移動はタイムスロットを基準として行い，各シミュレーションにおけるプレイヤの移動速度は 1 [マス/タイムスロット]，2 [マス/タイムスロット]，4 [マス/タイムスロット] とする．

– 相互干渉領域

相互干渉領域はプレイヤが異なる領域管理ノードが管理する部分領域付近に移動した際に，隣接する領域管理ノード間でプレイヤデータを共有することで，プレイヤがそこに移動する際にもゲームの整合性を維持するためのものである．プレイヤが異なる領域に移動する前に相互干渉領域内でプレイヤデータの共有を行う必要があるため，プレイヤと領域管理ノードが通信できるだけの相互干渉領域を設定しなければならない．本シミュレーションはプレイヤの移動速度をそれぞれ 1 [マス/タイムスロット]，2 [マス/タイムスロット]，4 [マス/タイムスロット] として行うため，相互干渉領域もそれぞれ 1 [マス]，2 [マス]，4 [マス] と設定する．

(5) 負荷

分割管理ノードの領域を分割する際に発生する計算負荷は通信負荷に比べて十分小さいものと仮定する．

– 管理負荷

領域管理ノード–プレイヤ間の通信負荷と相互干渉領域に対応する領域管理ノード–プレイヤ間の通信負荷の和を指す．通信内容はプレイヤの移動やチャットなどの操作により発生する行動情報の通信 (プレイヤ⇒領域管理ノード)，ゲーム状態をプレイヤに通知する通信 (領域管理ノード⇒プレイヤ)，プレイヤの状態や位置を領域管理ノードが管理する情報と同期する通信の 3 種類がある．クライアント–サーバ間で行われる上記 3 種類の通信負荷は最も高くなる状況下でプレイヤ 1 人あたり 31156[bps] であると，永塚正博氏 [15] らによる研究で述べられている．本論文の領域管理ノードは，部分領域内で局所的な C/S 型を構築する．よって，領域管理ノード–プレイヤノード間の通信負荷は永塚正博氏 [15] らの研究を参考にして，1 人のプレイヤと領域管理ノードが通信する際に発生する負荷を 30[kbyte] と設定する．また，一つの部分領域を作成する際に，分割管理ノードは既存の領域管理ノードと新しい部分領域の領域管理ノードの二つと通信を行う．よって，一つの部分領域を作成する際に発生する分割管理ノードの通信負荷は，プレイヤノードと領域管理ノードが通信する際に発生する負荷の倍の 60[kbyte] と設定する．

– 分割負荷

分割管理ノード–領域管理ノード間の通信負荷を指す．具体的には，分割管理ノードが新しい部分領域に領域管理ノードを割り当てる際に，領域管理ノードに部分領域情報と相互干渉領域情報を通信する負荷と領域の情報を書き換える際に発生する負荷の和を指す．

– 総負荷

管理負荷と分割負荷の和を指す．

本シミュレーションのパラメータを表 2 に示す．

本シミュレーションの評価項目を以下に示す．

(1) 部分領域数

仮想空間全体の部分領域数を測定する．

(2) 管理負荷

各領域管理ノードが受けるプレイヤの行動情報の通信負荷，ゲーム状態をプレイヤに通知する通信負荷，プレイヤの状態や位置を同期する通信負荷の和を測定する．

(3) 分割負荷

分割管理ノードが受ける通信負荷と分割時にマスの情報を書き換える処理負荷の和を測定する．

(4) 総負荷

シミュレーションで発生する負荷の合計値を測定する．

各シミュレーションでは，プレイヤは設定したホットスポットを中心に新規参入する．シミュレーションを 100 回行い，その平均をとったものを評価する．

5.2 シミュレーション結果

シミュレーション結果はシミュレーション時間 [s] を基準に示す．各グラフに記した灰色の縦線はプレイヤ

表 2　シミュレーションのパラメータ
Table 2　Parameters in simulation.

仮想空間全体	64 × 64 [マス]
ホットスポットの大きさ	7 × 7 [マス]
タイムスロット	0.5[s]
シミュレーション時間	5300[s]
分割しきい値	30[人]
結合しきい値	10[人]
領域プレイヤ数	20[人]
プレイヤの初期配置人数	200[人]
シミュレーション時間 1[s]～1800[s] のプレイヤ参入人数	2[/s]
シミュレーション時間 1[s]～1800[s] のプレイヤ離脱人数	1[/s]
シミュレーション時間 1801[s]～3600[s] のプレイヤ参入人数	0[/s]
シミュレーション時間 1801[s]～3600[s] のプレイヤ離脱人数	0[/s]
シミュレーション時間 3601[s]～5300[s] のプレイヤ参入人数	0[/s]
シミュレーション時間 3601[s]～5300[s] のプレイヤ離脱人数	1[/s]
分割管理ノードと領域管理ノード間の通信間隔	0.5[s]
プレイヤノードと領域管理ノード間の通信間隔	0.5[s]
1 マスの情報を書き換える処理負荷	1[byte]
1 人のプレイヤと領域管理ノードの通信負荷	30[kbyte]
一つの部分領域作成時の分割管理ノードの通信負荷	60[kbyte]

の新規参入が止まる 1800[s] とプレイヤ数の定常状態が終わる 3600[s] を表している．

プレイヤの移動速度が 1 [マス/タイムスロット]，2 [マス/タイムスロット]，4 [マス/タイムスロット] で通常移動するときの管理負荷を図 13～15 に示す．

図 13～15 からプレイヤの移動速度の増加に伴って管理負荷が増加することがわかる．これは，プレイヤの移動速度が速くなるにつれて，相互干渉領域に移動する頻度が増加し，相互干渉領域に対応する領域管理ノード–プレイヤ間の通信負荷が増加したことが原因と考えられる．領域管理ノード–プレイヤ間の通信負荷はプレイヤ数にのみ影響され，提案アルゴリズムと従来法のプレイヤ数は同じように増加，減少することから，提案アルゴリズムと従来法の領域管理ノード–プレイヤ間の通信回数の総和は等しい．よって，図 13～15 の管理負荷の差は，相互干渉領域に対応する領域管理ノード–プレイヤ間の通信負荷の差を示している．また，どの移動速度でも提案アルゴリズムは従来法より管理負荷が軽減できていることがわかる．

以降の実験結果は全てプレイヤの移動速度を 2 [マス/タイムスロット] としたものである．

提案アルゴリズムでのプレイヤの通常移動とジグザグ移動の管理負荷を図 16 に示し，提案アルゴリズムと従来法でのプレイヤのジグザグ移動時の管理負荷を図 17 に示す．

図 16 より，ジグザグ移動時の管理負荷は通常移動時に比べて，プレイヤ数の増加時には早く増加し，プレイヤ数の減少時には早く減少することがわかる．通常移動ではホットスポットに向かって移動する確率は 1/10 だが，ジグザグ移動では全てのプレイヤが定められたホットスポットに向かって移動する．そのため，ジグザグ移動ではプレイヤのほとんどがホットスポット，またはその周辺に存在し，早い段階でプレイヤの密集が発生する．プレイヤの密集箇所では小さな部分領域が多数形成され，相互干渉領域内に存在するプレイヤ数も増えるため相互干渉負荷が増加する．よって，ジグザグ移動時の相互干渉負荷が通常移動時に比べて早く増加したと考えられる．また，ジグザグ移動時のプレイヤ数の減少はプレイヤの密集の解消に直結するため，相互干渉負荷が通常移動時に比べて早く減少したと考えられる．プレイヤ数の定常時にはジグザグ移動時の相互干渉負荷が通常移動時に比べて大きくなっているが，これはプレイヤが相互干渉領域を縫うように移動したため相互干渉負荷が大きくなったと考えられる．

提案アルゴリズムではプレイヤの移動パターンにより管理負荷の増減速度が若干変化するが，どちらの移動パターンでも従来法以上に相互干渉負荷を軽減できていることが図 14，17 からわかる．また，四分木，JoHNUM は提案アルゴリズムとは異なり移動パターンによる管理負荷の変化があまりみられない．この理由として，四分木，JoHNUM のどちらの手法もホットスポットの位置に関係なくあらかじめ決められた形に領域を分割し部分領域を作成するため，プレイヤの移動パターンの変化による影響が小さかったと考えられる．

以降の実験結果は全てプレイヤの移動が通常移動としたものである．

次に各手法における部分領域数を図 18，分割負荷を図 19 に示す．

図 18 より，提案アルゴリズムは従来法に対して最大半分程度まで部分領域数を削減することに成功している．これは管理ノード数を削減できることから P2P 型 MMORPG にとって望ましい結果である．また，四分木と JoHNUM はプレイヤ数が減少しても部分領域を結合できないために部分領域数を削減できておらず，

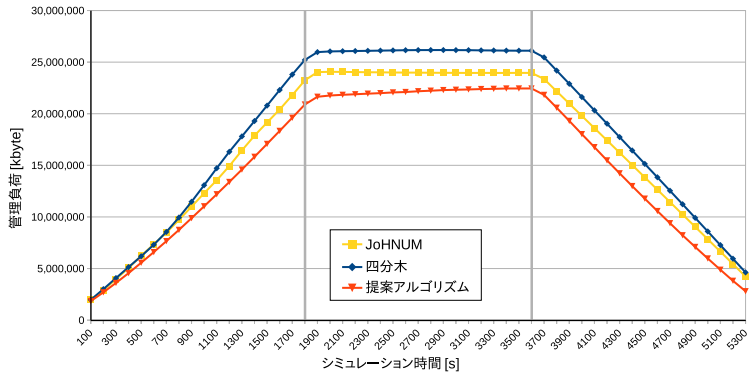

図 13　移動速度が 1 [マス/タイムスロット] のプレイヤに対する管理負荷
Fig. 13　The management load such that the player's moving speed is 1[mass/time slot].

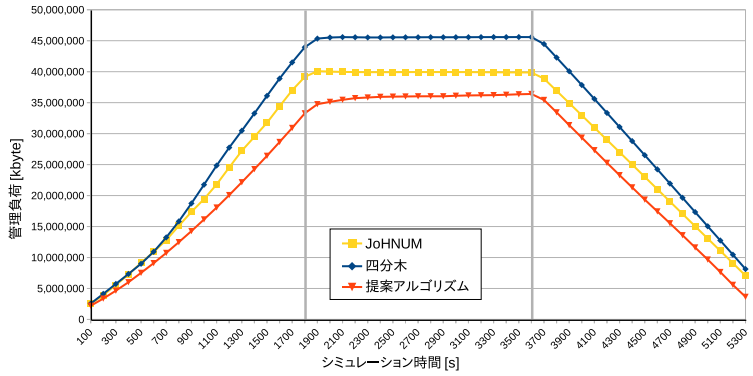

図 14　移動速度が 2 [マス/タイムスロット] のプレイヤに対する管理負荷
Fig. 14　The management load such that the player's moving speed is 2[mass/time slot].

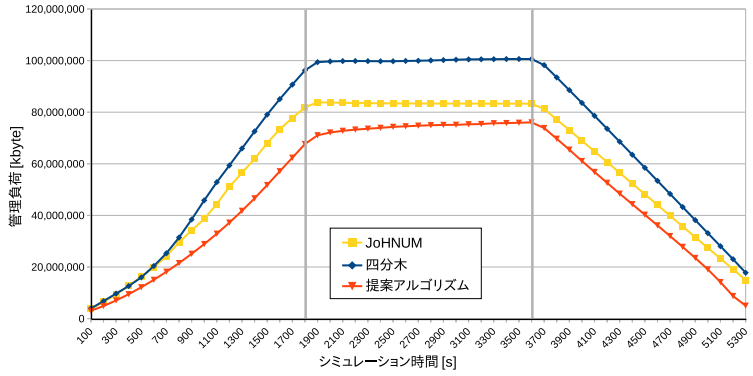

図 15　移動速度が 4 [マス/タイムスロット] のプレイヤに対する管理負荷
Fig. 15　The management load such that the player's moving speed is 4[mass/time slot].

提案アルゴリズムは共に余分な部分領域の結合が可能なためプレイヤ数の減少に動的に対応できていることがわかる．一方，提案アルゴリズムはプレイヤ数の増加と減少の両方に動的に対応するため，部分領域情報の更新回数が多く分割負荷は従来法より大きくなっていることが図 19 よりわかる．また，提案アルゴリズ

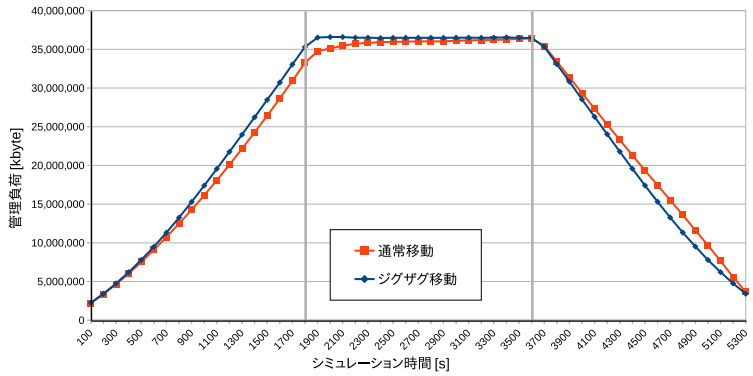

図 16　提案アルゴリズムでのプレイヤの移動に対する管理負荷
Fig. 16　The management load of the proposed algorithm by the movement.

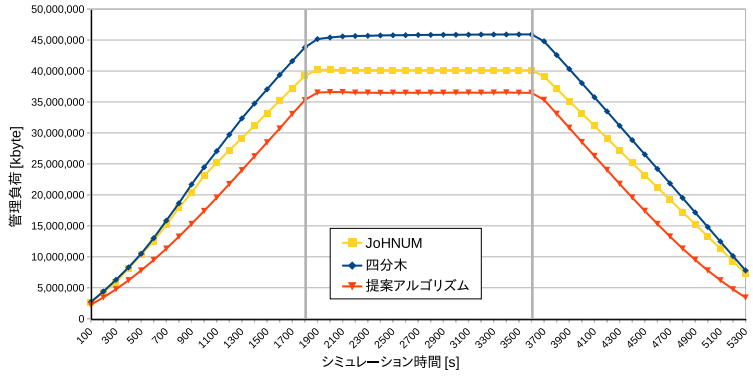

図 17　プレイヤのジグザグ移動に対する管理負荷
Fig. 17　The management load by the zigzag movement.

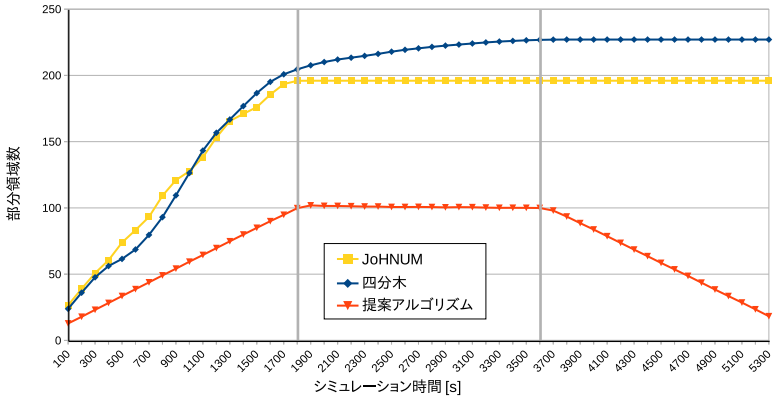

図 18　部分領域数
Fig. 18　Number of regions.

ムはプレイヤ数の増加中に分割負荷が増加し，その後頭打ちして減少していく．

プレイヤ数の増加時は，ホットスポット内の部分領域でプレイヤの密集が発生しやすくなり，部分領域の分割が高い頻度で発生する．また，ホットスポット以外の部分領域では過疎状態になりやすく部分領域の結合も発生する．そのため，プレイヤ数の増加時に最も分割負荷が大きくなったと考えられる．また，プレイ

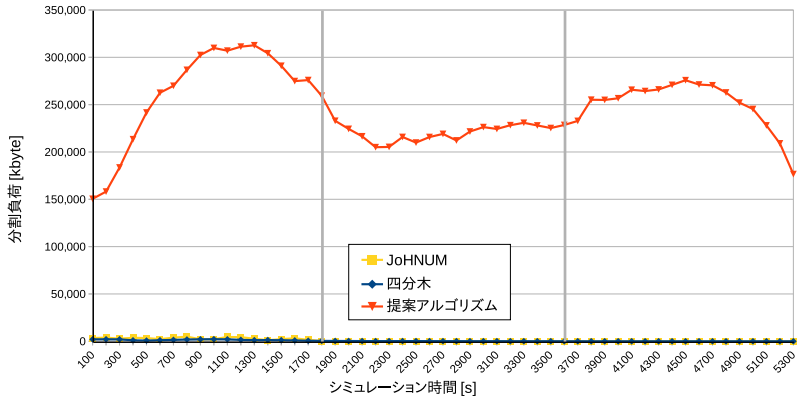

図 19　分 割 負 荷
Fig. 19　The division load.

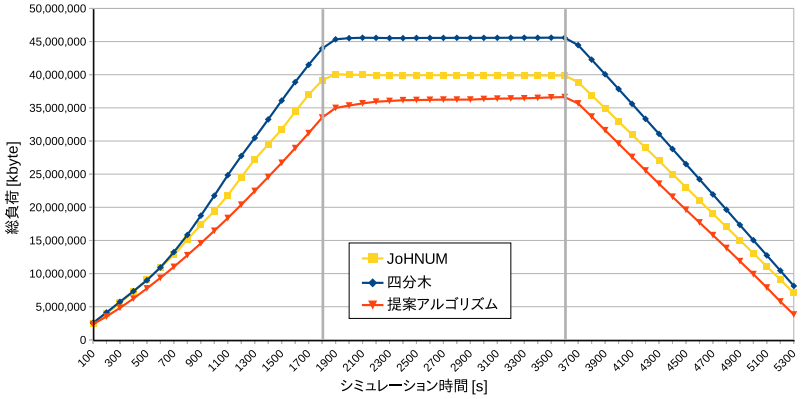

図 20　総　負　荷
Fig. 20　The total load.

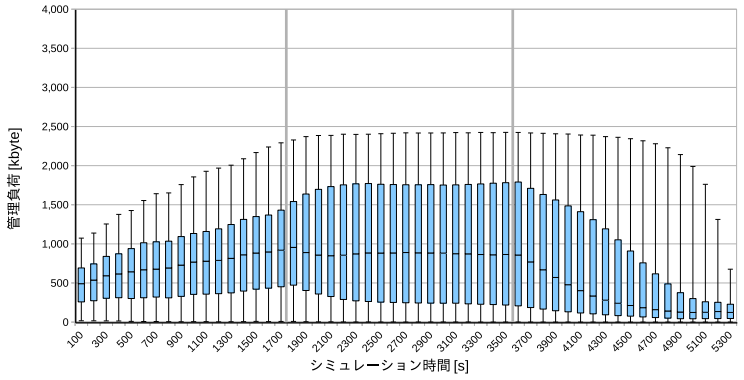

図 21　四分木における各領域管理ノードにかかる管理負荷の箱ひげ図
Fig. 21　Box plot diagram of the management load on each area management node in the quadtree.

ヤ数の定常時はプレイヤ数の増加時にある程度ホットスポットに対応した部分領域が作成されているため，分割・結合の発生頻度が低く分割負荷は一定の値になったと考えられる．プレイヤ数の減少時は部分領域の結合の発生頻度は高くなるが部分領域の分割の発生頻度が低く，プレイヤ数の増加時に比べて分割負荷は低く

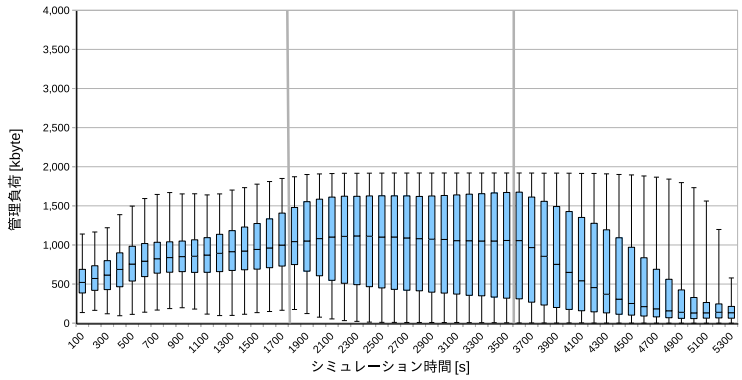

図 22　JoHNUM における各領域管理ノードにかかる管理負荷の箱ひげ図
Fig. 22　Box plot diagram of the management load on each area management node in the JoHNUM.

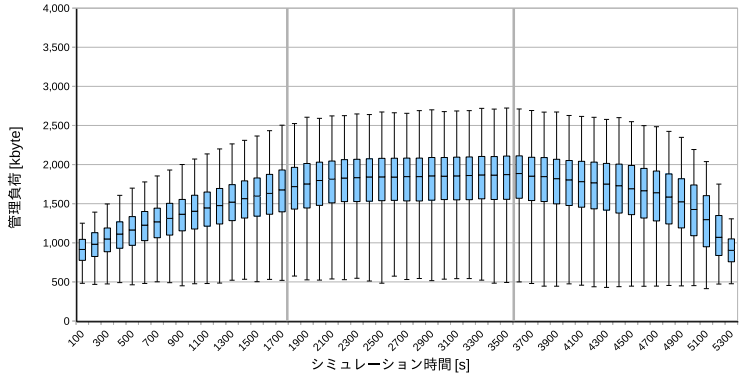

図 23　提案アルゴリズムにおける各領域管理ノードにかかる管理負荷の箱ひげ図
Fig. 23　Box plot diagram of the management load on each area management node in the proposed algorithm.

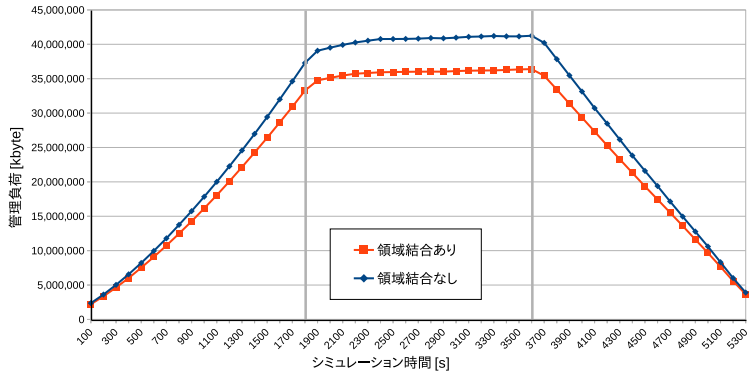

図 24　提案アルゴリズムにおける領域結合の有無に対する管理負荷
Fig. 24　The management load by area join in the proposed algorithm.

なったと考えられる．

次に各手法に対する総負荷を図 20，各手法において領域管理ノードが 1 タイムスロット当たりに受ける管理負荷の箱ひげ図を図 21～23 に示す．

図 20 より，提案アルゴリズムが従来法より負荷を軽減できており，図 14，19，20 を比較すると総負荷

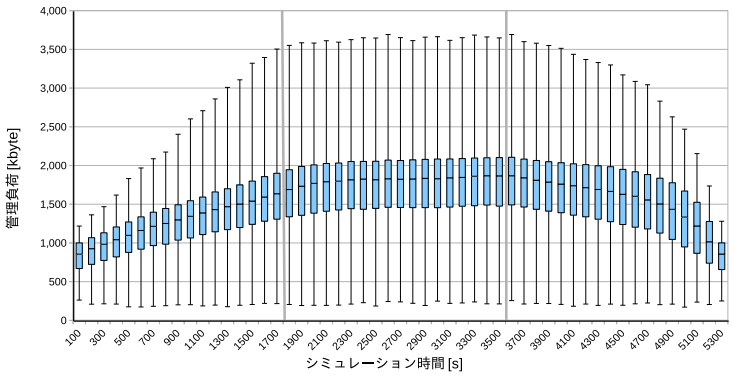

図 25　領域結合を行わない提案アルゴリズムにおける各領域管理ノードにかかる管理負荷の箱ひげ図

Fig. 25　Box plot diagram of the management load on each area management node in the proposed algorithm without the area combining.

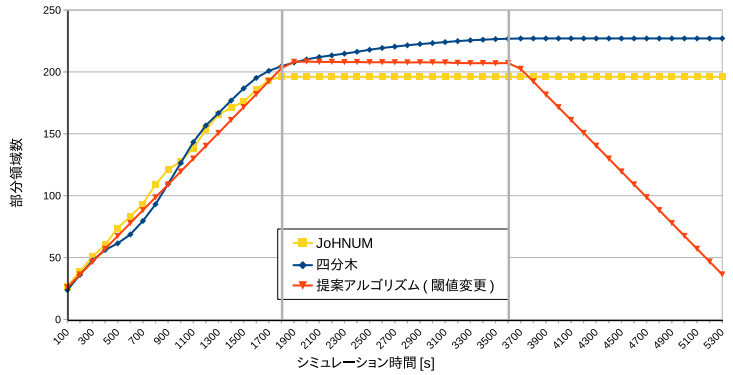

図 26　しきい値変更後の提案アルゴリズムの部分領域数

Fig. 26　Number of regions in the proposed algorithm of threshold change.

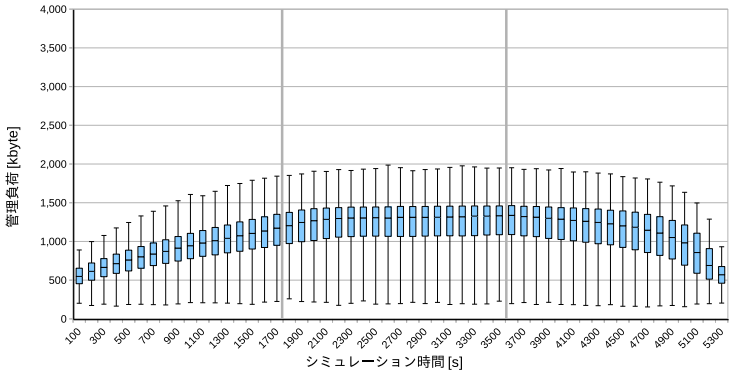

図 27　しきい値変更後の提案アルゴリズムの各領域管理ノードにかかる管理負荷の箱ひげ図

Fig. 27　Box plot diagram of the management load on each area management node in the proposed algorithm of threshold change.

の中の分割負荷が占める割合は最大 0.6%程度ととても小さく，負荷の大半は管理負荷であることがわかる．また，図 21〜23 より，従来法に比べて提案アルゴリズムは各領域管理ノードに均等に負荷を割り当てていることがわかる．

次に提案アルゴリズムでの領域結合の有無による管

理負荷を図 24，領域管理ノードが 1 タイムスロット当たりに受ける管理負荷の箱ひげ図を図 25 に示す．

領域結合を行うことで図 24 から管理負荷を軽減でき，図 23，25 から各領域管理ノードにかかる負荷の最大値を軽減できることがわかる．この結果から，細長い部分領域を隣接する部分領域と結合させる領域結合は有用であるといえる．

改めて図 21〜23 を見てみると，提案アルゴリズムは従来法より各領域管理ノードにかかる負荷値が全体的に高くなっていることがわかる．これは，提案アルゴリズムは従来法に比べて部分領域数すなわち管理ノード数が少ないためであると考えられる．そこで，提案アルゴリズムの分割しきい値を 15，結合しきい値を 5，領域プレイヤ数を 10 に変更し，部分領域数を従来法とほぼ等しい数まで増加させた場合の領域管理ノードが 1 タイムスロット当たりに受ける管理負荷の箱ひげ図を求める．従来法と各しきい値変更後の提案アルゴリズムの部分領域数を図 26 に示し，各しきい値変更後の提案アルゴリズムにおける領域管理ノードが 1 タイムスロット当たりに受ける管理負荷の箱ひげ図を図 27 に示す．

図 27 より図 21，22 と比較して，部分領域数を従来法とほぼ等しい数まで増加させた場合，提案アルゴリズムは各領域管理ノードに従来法より均等に分散できていることがわかる．よって，提案アルゴリズムは負荷の分散を達成しつつ，従来法より負荷の軽減と均等な負荷の分散ができることがわかる．

6. むすび

本論文では，MMORPG においてプレイヤの移動を考慮し動的に部分領域の分割・結合を行う動的領域分割結合アルゴリズムを提案した．また，シミュレーションによって動的領域分割結合アルゴリズムが従来法に比べて負荷の軽減と均等な負荷の分散ができているかを評価した．シミュレーション結果より，動的領域分割結合アルゴリズムは，1 台の管理ノードの負荷が増加しているものの，管理ノード数を削減でき，総負荷の軽減と均等な負荷の分散ができていることを示した．よって，動的領域分割結合アルゴリズムは有用であるといえる．

今回，分割管理ノードと領域管理ノードは，離脱しないものと仮定してシミュレーション実験を行った．分割管理ノードと領域管理ノードの離脱によるデータの損失を避けるためにはバックアップノードを考慮する必要があり，この点については今後の課題である．

謝辞　貴重なご意見とご指摘をくださった査読者の方々に深謝する．本研究の一部は，JSPS 科研費 25330123 の助成を受けたものである．ここに記して謝意を表す．

文　献

[1] Blizzard Entertainment: World of warcraftm, Blizzard Entertainment, http://www.worldofwarcraft.com/index.xml/ (accessed 2014-09-11)

[2] D. Philippe and V. Ariel, “Improving scalability in MMOGs — ScalaMo: A new architecture,” Cite SeerX, http://scalamo.sourceforge.net/slides.pdf (accessed 2014-09-11)

[3] U. Farooq and J. Glauert, “Joint hierarchical nodes based user management (JoHNUM) infrastructure for the development of scalable and consistent virtual worlds,” 2009 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, pp.105–112, 2009.

[4] B. De Vleeschauwer, B. Van Den Bossche, T. Verdickt, F. De Turck, B. Dhoedt, and P. Demeester, “Dynamic microcell assignment for massively multiplayer online gaming,” Proc. 4th ACM SIGCOMM workshop on Network and system support for games - NetGames ’05, pp.1–7, 2005.

[5] R. Thawonmas, M. Kurashige, and K.T. Chen, “Detection of landmarks for clustering of online-game players,” Int. J. Virtual Reality, pp.11–16, 2007.

[6] D.T. Ahmed and S. Shirmohammadi, “A dynamic area of interest management and collaboration model for P2P,” Proc. 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications, pp.27–34, 2008.

[7] 遠藤　伶，高木健士，北　望，重野　寛，“MMO 仮想環境におけるユーザ密度の変化に対応した負荷分散手法,” (セッション B-8:P2P・オーバーレイネットワーク (2))，情処学研報，CSEC, pp.213–218, 2008.

[8] S.-Y. Hu and G.-M. Liao, “Scalable Peer-to-Peer networked virtual environment,” NetGames ’04 Proc. 3rd ACM SIGCOMM Workshop on Network and System Support for Games, pp.129–133, 2004.

[9] S.-Y. Hu, S.-C. Chang, and J.-R. Jiang, “Voronoi state management for Peer-to-Peer massively multiplayer online games,” Consumer Communications and Networking Conference, 2008, CCNC 2008, 5th IEEE, pp.1134–1138, 2008.

[10] M. Assiotis and V. Tzanov, “A distributed architecture for MMORPG,” NetGames ’06 Proc. 5th ACM SIGCOMM Workshop on Network and System Support for Games, pp.73–78, 2006.

[11] 山崎孝裕，金田憲二，大山恵弘，米澤明憲，“柔軟性と拡張性を備えた大規模多人数オンラインゲームのための枠組み,” 信学技報，CPSY, コンピュータシステム，vol.105, no.226, pp.61–66, 2005.

[12] 小林基成，鈴木俊博，永田智大，カーンアシック，趙　晩熙，“オンラインゲームのための仮想ネットワークの資源割り当て方法の検討，” 信学技報，NS2008-13, 2008.
[13] S. Yamamoto, Y. Murata, K. Yasumoto, and M. Ito, “A distributed event delivery method with load balancing for MMORPG,” NetGames ’05 Proc. 4th ACM SIGCOMM Workshop on Network and System Support for Games, pp.1–8, 2005.
[14] Passion For The Future.net: Passion For The Future.net, http://www.ringolab.com/note/daiya/archives/001278.html, (accessed 2014-09-11)
[15] 永塚正博，山名早人，P2P を用いた多人数同時参加型オンライン RPG (MMORPG) の実現，Bachelor’s degree (School of Science and Engineering)（早稲田大学 理工学部），2004.

（平成 26 年 3 月 25 日受付，9 月 19 日再受付）

榎原　博之　（正員）

1982 阪大・工・通信卒．1987 同大大学院博士 (通信) 課程了．同年阪大工学部助手．1994 関西大工学部専任講師となり，現在，准教授．組合せ最適化問題，計算幾何学，並列アルゴリズム等の研究に従事．工博．情報処理学会，IEEE，ACM 各会員．

吉岡　　啓

1989 年生．2012 年関西大学システム理工学部電気電子情報工学科卒業．2014 年同大学大学院理工学研究科博士課程前期課程システムデザイン専攻修了．現在，NEC ソリューションイノベータ株式会社に勤務．在学中負荷分散アルゴリズムの研究に従事．

松崎　頼人　（学生員）

1988 年生．2011 年関西大学システム理工学部電気電子情報工学科卒業．2013 年同大学大学院理工学研究科博士課程前期課程システムデザイン専攻修了．同年より同大学大学院理工学研究科博士課程後期課程総合理工学専攻在籍．災害時緊急ネットワーク構築の研究に従事．電子情報通信学会学生会員．