「マルチメディア，分散，協調とモバイル
(DICOMO2017) シンポジウム」 平成29年6月

エクスターナルグリッドにおける網羅法の処理ノード数増加に対する抑制手法の提案

山口 晃右[1] 藤橋 卓也[1] 遠藤 慶一[1] 小林 真也[1]

概要：ネットワークに接続された計算機に処理を分散配置し，安価に高性能な計算資源を獲得するための分散処理技術として，グリッドコンピューティングが知られている．グリッドコンピューティングの中でも，インターネットに接続された計算機を計算資源に利用するものを，エクスターナルグリッドと呼ぶ．インターネットに接続された計算機は無数に存在し，エクスターナルグリッドの利用者は，実質無制限に計算資源を獲得できる．しかし，インターネットに接続された計算機の所有者の中には，不正行為を働こうとする者がいる．このような悪意のある所有者による，エクスターナルグリッドに対する不正行為のリスクを低減する諸技術として，我々はセキュアプロセッシングの研究を行っている．同時に，エクスターナルグリッドを高速化する手法も幾つか提案し，その内の一つに網羅法がある．この網羅法は，安定したエクスターナルグリッドの高速化を実現できるが，計算に参加する計算機の利用数が著しく増加し，エクスターナルグリッドの安全性を低下させる原因となっている．本稿の目的は，網羅法に比べて安全性の高いエクスターナルグリッドの高速化を実現することである．到達目標として，網羅法による網羅法の欠点を補った，エクスターナルグリッドに対する新たな高速化手法を提案する．更に，先行研究で提案された高速化手法と比較して，本稿での提案手法の性能の比較・評価を行い，エクスターナルグリッドの性能向上に寄与することを示す．

1. はじめに

グリッドコンピューティングの一種に，エクスターナルグリッドがある．エクスターナルグリッドは，インターネットに接続された不特定多数の計算機を計算に利用することで，高い計算能力を獲得できる．このとき，エクスターナルグリッドに依頼されるプログラムを，実際に処理する計算機を “ノード” と呼ぶ．

しかし，インターネットに接続された不特定多数の計算機の中には，悪意を持った管理者が保有している計算機が混入する可能性がある．この悪意を持った管理者を “悪人” と呼ぶ．また，悪人が所有する計算機を “悪人ノード” と呼ぶ．

悪人ノードによる不正行為に対する対策として，我々はセキュアプロセッシングを提案している．セキュアプロセッシングは，不正行為によるリスクを軽減する手法の集合である．また，エクスターナルグリッドの処理を高速化する手法も提案している．この高速化手法の一つが，“網羅法” である．

網羅法は，エクスターナルグリッドの高速化を実現できる一方で，エクスターナルグリッドを構成するノード数が著しく増えるという短所を持っている．ノード数の増加は，悪人ノードの混入数の増加につながり，不正行為のリスクが高まる．

本研究の目的は，エクスターナルグリッドの安全性と性能向上を目指すことである．到達目標として，網羅法を利用したエクスターナルグリッドのノード数の増加を抑制する手法を提案し，悪人による不正行為のリスクの軽減に繋がることを示す．具体的には，先行研究で提案された高速化手法である “閾値暫定法” や “網羅法” と比較して，エクスターナルグリッドの計算に参加するノードの数の増減，エクスターナルグリッドの処理が終了するまでの時間などを比較することで，本手法の特徴と効果を示す．

2. エクスターナルグリッドとセキュアプロセッシング

2.1 エクスターナルグリッドの現状

エクスターナルグリッドは，インターネット上の計算機を計算資源として利用することで，高い処理性能を実現する．エクスターナルグリッドは，大規模演算やシミュレーションなどの領域での利用が期待されている．例えば，エクスターナルグリッドを実現するオープンソフトウェアとして BOINC[1] がある．また，BOINC を利用し

[1] 愛媛大学大学院理工学研究科

た SETI@home[2] などのプロジェクトが存在する．

しかし，今日に至るまでエクスターナルグリッドを利用した営利目的のサービスの成功は確認されていない．原因は，エクスターナルグリッドに処理を依頼した際に，処理内容の搾取や処理結果の真正性が保証されていないためである．インターネット上の処理ノードは多数の管理者によって管理されている．管理者の中には，悪人が一定数存在すると考えられる．我々は，この悪人が管理する悪人ノードによる不正行為に対する対策として，セキュアプロセッシング [3] の研究を進めている．

2.2 セキュアプロセッシング

セキュアプロセッシングは，悪人ノードによる不正行為のリスクを軽減するための技術の総称である．悪人ノードが行う不正行為には，“不正な解析” と “改竄” の 2 種類がある．

不正な解析

悪人が，エクスターナルグリッドに依頼された処理の内容に含まれる情報を盗み取るための不正行為である．

改竄

悪人が，エクスターナルグリッドに依頼された処理の結果を誤ったものにする不正行為である．

セキュアプロセッシングでは，上記の不正な解析と改竄に対する対策として，それぞれ “プログラム分割” と “処理の多重化” がある．

プログラム分割

エクスターナルグリッドに依頼されたプログラムを分割する．分割された小さなプログラムを “プログラム断片” と呼ぶ．元のプログラムを分割したて得られたプログラム断片の数を “分割数” と呼ぶ．各処理ノードは，このプログラム断片を実行する．悪人ノードにプログラム断片が渡ったとしても，元のプログラムの一部であるので，悪人ノードが取得できる情報量は限られたものになる．このように，悪人ノードによる不正な解析に対して効果がある．

処理の多重化

1 つのプログラム断片を 1 台の処理ノードに実行させるのみでは，断片を実行したノードが改竄を行う悪人ノードである場合，以降の処理が誤った実行結果を元にしたものとなってしまう．

処理の多重化では，1 つのプログラム断片を複数台の処理ノードに実行させる．このときの処理ノードの台数を “多重度” と呼ぶ．そして，多重度台の処理ノードが返した実行結果で投票を行い，過半数を獲得した実行結果を用いて，以降の処理を進める．悪人の存在確率が，0.5 未満であれば，多重化を行うことで，正しい結果を得る確率が向上する．

投票を行う多重度台の処理ノードのまとまりを “ブロック” と呼ぶ．投票の際に，過半数に達する実行結果が一つも無い場合には “票割れ” を起こす．この処理の多重化によって，1 つのプログラム断片の実行結果の信頼性を高め，改竄のリスクを軽減する．

3. 既存の高速化手法

3.1 先行処理

先行処理 [4] とは，処理の多重化が利用されたエクスターナルグリッドの高速化を実現するための手法である．

処理の多重化によって，各ブロックから次のブロックに処理が移行するには，投票の結果を待つ必要がある．このため，1 つのブロックの処理時間は，ブロック内で多数決を決する投票したノードの処理時間である．

図 1 は，処理の多重化による処理時間と処理過程を表したものである．ノード 01 から 05 は，同一ブロックに属しており，同種のプログラム断片を処理する．図 1 のグラフの色は，実行結果の種類を表している．例えば，ノード 01，02 は同じ実行結果を返している．

図 1 において，ブロック 01 で多数決を決する投票をしたのは，ノード 03 である．よって，ブロック 01 の処理時間は T_1 である．T_1 の後に，ブロック 02 の処理が開始される．

一方，先行処理では，投票を待つことなく，ブロック内で最早の実行結果を元にして処理を進めるという処理を行う．よって，投票確定以前に後続処理を開始することができるので，各ブロックの処理時間の短縮化ができ，プログラム全体の処理時間の短縮化を実現できる．

図 2 は，先行処理を用いた場合の処理を時系列順に並べたものである．

先行処理を用いている図 2 では，ブロック 02 の生成時刻が，T_1 よりも早い T_2 であることがわかる．更に，T_1 以前に，T_3 でブロック 03 での処理が開始されている．以上のように，ブロック内で最早の実行結果を用いることで，

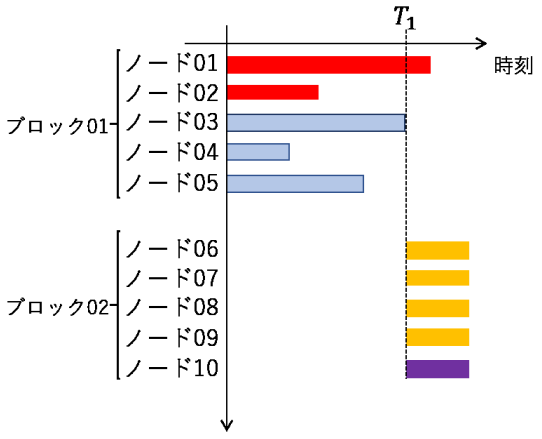

図 1 処理の多重化による処理時間

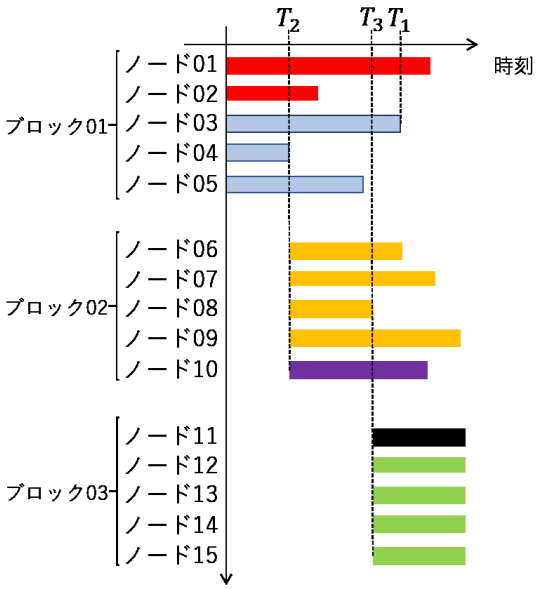

図 2　先行処理による高速化の仕組み

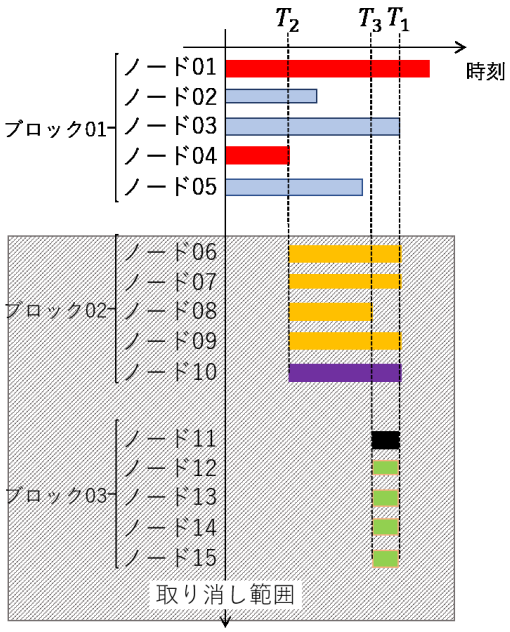

図 3　ロールバックの発生過程

処理時間を短くすることができる．

しかしながら，この最早の実行結果が正しくない場合がある．

図 3 は，ロールバックが発生するようなエクスターナルグリッドの処理過程を表している．

図 3 のブロック 01 では，時刻 T_2 にて，ノード 04 が最早の実行結果を出し，ブロック 02 の処理が開始される．そして，時刻 T_1 にて，ブロック 01 の投票が完了し，ノード 02 の実行結果が誤りであったことがわかる．この時，唯一先行している処理である，ノード 06 からノード 15 までの処理が取り消される．そして，時刻 T_1 から，再びブロック 02 で処理していたプログラム断片の処理を開始する．つまり，先行して行っていな処理は全て無駄であったことになる．

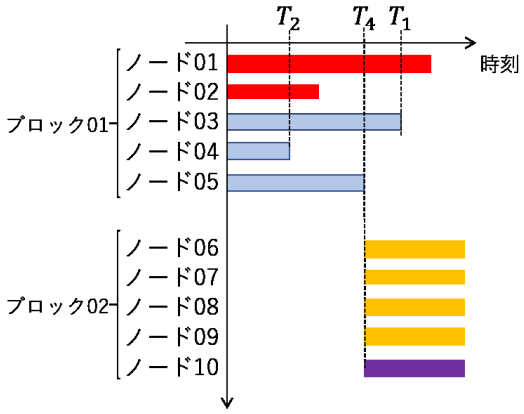

図 4　閾値網羅法のブロック生成過程と処理時間

このように，誤った最早の実行結果を元にして先行していた処理が取り消される現象を，“ロールバック”と呼ぶ．このロールバックによって，先行処理による高速化が損なわれる．

3.2　閾値暫定法

閾値暫定法とは，先行処理におけるロールバックの発生頻度を抑えるために考案された手法である．ブロック内で最早の実行結果を用いるのではなく，“閾値”と同数の実行結果が集まった後に先行処理を行う．つまり，確定には至らないが，見込みとして正しい結果である事の期待が大きい結果が得られた後に，先行処理を行う．これにより，ロールバックの発生を抑制することが期待できる．

暫定閾値法を用いた場合の処理の時系列を図 4 に示す．閾値は 2 である．

図 3 と異なり，図 4 では，ノード 04 の最早の実行結果を利用してブロック 02 へ処理を先行させるのではなく，ノード 05 の処理が完了した時刻 T_4 にてブロック 02 に処理が移行している．これは，ノード 02 の実行結果とノード 05 の実行結果が同じであり，同じ実行結果が閾値と同数確認されたため，先行処理を行ったためである．

閾値以上の同一の実行結果を得られた後に先行処理を開始することで，先行処理に用いる暫定的な結果の確からしさを高めることができる．しかしながら，確定以前に先行処理を開始することになるので，ロールバックの発生をなくすことはできない．

また，図 4 において T_4 は，処理の多重化におけるブロック 02 の開始時刻 T_1 より早く，図 2 の T_2 より遅いということがわかる．処理時間に関して，閾値暫定法は，処理の多重化よりも短縮化ができるが，最早の実行結果が正しい場合の先行処理の処理時間よりも長くなるといえる．

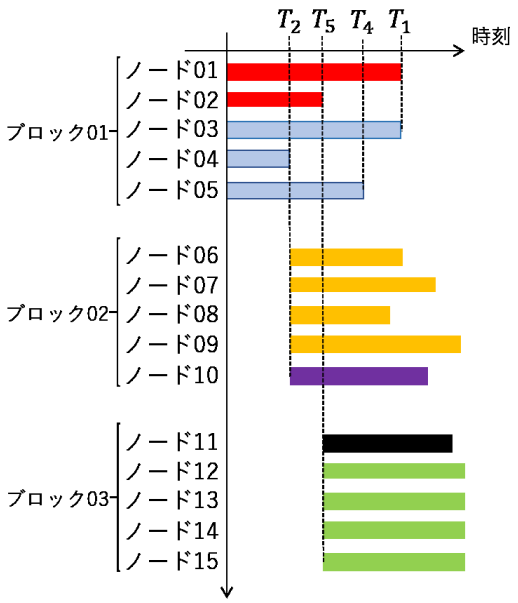

図 5　網羅法のブロック生成過程と処理時間

3.3　網羅法

網羅法では，ノードが所属するブロック内で新規の実行結果が出力された際，出力される毎に，新しく多重度と同数のノードを選出し，処理を進める．そのため，ロールバックの発生は完全に防ぐことができる．

図 5 は，網羅法によるブロックの生成過程と，処理時間の関係を示している．

図 5 のように，これまでに得られたものとは異なる実行結果が得られる度に，新たなブロックが生成される．そのため，実行結果の種類が多い場合などは，ノード数が著しく増える．この現象は，悪人が集団を形成して，各悪人ノードがそれぞれ異なる実行結果を返す，という振る舞いをした際に，顕著に現れる．エクスターナルグリッドを構成するノード数の増加は，悪人ノードの混入数の増加につながり，不正行為に関するリスクが高まる．

また，図 5 より，ブロック 02 の処理の開始時刻である T_2 は，処理の多重化や閾値暫定法におけるブロック 02 の処理開始時刻である T_4 や T_1 よりも早い．そのため，網羅法は処理の多重化や閾値暫定法より，依頼されるプログラムの処理の高速化が実現できる．更に，網羅法では先行処理におけるロールバックが発生しないので，悪人が存在するエクスターナルグリッドにおいて，網羅法は先行処理よりも処理時間の短縮化が期待できる．

3.4　各高速化手法のまとめ

3 節内で述べた高速化手法について，利用ノード数と処理時間の観点で，以下のようにまとめた．

利用ノード数

利用ノード数が最小で済む高速化手法は，処理の多重化であるといえる．次いで，閾値暫定法，先行処理，網羅法の順に利用処理ノード数が多くなる．　処理の多重化は，ロールバックも網羅的な先行処理も行わないので，利用ノード数は最小で済む．先行処理や閾値暫定法は，ロールバックが発生した際にノードを再選定するので，その分のノード数が増える．網羅法は，処理が網羅的に先行していくので，前述した高速化手法の中では，最も利用ノード数が多くなる．

処理時間

処理時間に関して，網羅法が最短で処理を完了させることができる．閾値暫定法は，ロールバックの発生頻度が低い分，先行処理より高速に処理ができる．一方で，ロールバックが発生しない場合は，閾値暫定法より先行処理の方が，高速化の効果が大きい．いずれの高速化手法も，処理の多重化よりも処理時間は短くなることが期待できる．

以上のように，網羅法は他の手法より高速化の効果が大きい反面，利用ノード数が多くなるという問題がある．また，閾値暫定法は，網羅法ほどの安定した高速化が行えないが，利用ノード数を比較的抑えることができる．特に，利用ノード数が大きくなることは，エクスターナルグリッドに混入する悪人ノードの台数の増加につながり，不正行為のリスクの増大に繋がる．

4.　閾値網羅法

本稿では，網羅法によるロールバック抑制の効果を活かしつつ，ノードの増加を抑制することを目論み，閾値暫定法の実行結果が正しいという見込みを高める特徴を取り入れた，“閾値網羅法” を提案する．

網羅法は，新出の実行結果が 1 つ出るたびに，その結果を用いた先行処理を開始している．閾値網羅法は，同種の実行結果が “閾値” と同数集まるまで待ち，閾値以上となった場合に，網羅法と同様の先行処理を行う．ただし，閾値は，多重度の過半数より小さい値である．この手法は，閾値暫定法とは異なり，網羅法の一種であるので，ロールバックが発生しない．

図 6 は，閾値網羅法の挙動を表している．例えば，図 6 における閾値は，2 である．設定した閾値により，ノード 05 の実行が完了して，ブロック 02 が生成される．このように，図 6 では図 5 と異なり，新出の実行結果が一定数揃ってから，先行処理が開始している．そのため，図 6 では，網羅法を用いている図 5 よりも，時刻 T_1 の時点で利用ノード数が 1 ブロック分少なくなる．

閾値が大きいほど，網羅的に行う先行処理の数が少なくなると考えられるので，エクスターナルグリッドにおけるノード数は，単純な網羅法に比べて少なくなると考えられる．一方で，閾値が多重度の値に近づくほど，網羅法による高速性が失わると思われる．

処理時間に関しては，閾値網羅法はロールバックが発生

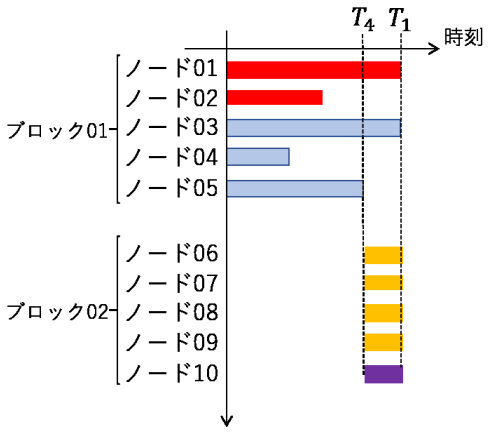

図 6　閾値網羅法のブロック生成過程と処理時間

しないため，低確率ながらもロールバックが発生する閾値暫定法より，高速化することができると考えられる．

5. 評価

5.1 評価指標

閾値網羅法のエクスターナルグリッドの性能を評価するために，以下の 3 つの指標を定める．

エクスターナルグリッドの安全性

エクスターナルグリッドの安全性を求めるため，エクスターナルグリッドに利用されるノードの総数を用いる．利用ノード数を比較することは，悪人による不正行為のリスクを推し量ることを意味する．悪人ノードには，各プログラム断片が配布される恐れがあり，悪人ノードの絶対数が多いことは，より悪人に情報が漏れてしまうことを示している．つまり，エクスターナルグリッド内に混入する悪人ノードの数が少ない方が，不正行為のリスクが少ないといえる．

悪人集団による不正解析のリスク

悪人集団による不正解析のリスクを評価するために，その悪人集団が取得できた，エクスターナルグリッドに依頼されるプログラムのプログラム断片の割合を利用する．悪人集団は，より多くのプログラム断片を取得することで，不正解析の難易度を下げることができる [5]．取得されたプログラム断片の割合が少ないほど，悪人集団による不正解析のリスクが低いことを表している．

エクスターナルグリッドの処理時間

高速化手法である閾値網羅法，閾値暫定法，網羅法の高速化の効果を測るため，エクスターナルグリッドの最終的な処理が完了するまでの処理時間を用いる．当然，処理時間に関してはより短いほうが良い．

以上で述べた値を，閾値暫定法や網羅法，閾値網羅法を利用したエクスターナルグリッドのシミュレーションにより求める．

5.2 シミュレーション条件

シミュレーションの条件は，以下の通りである．

モデルを特徴づけるパラメータ

シミュレーションモデルを特徴づけるパラメータである，プログラム分割数，多重度，悪人の存在確率，閾値網羅法における閾値，の 4 つを与える．

依頼されるプログラム

プログラムのサイズは 100 である．また，プログラム分割によって生成される断片は，連続した依存関係で繋がっている．1 つの断片のサイズは，プログラムサイズをプログラム分割数で等分した値である．

ノード数

エクスターナルグリッドに利用可能なノード数は，無数に存在する．

ノードの処理性能

ノードの処理性能は，単位時間あたりにノードが処理できるプログラムサイズであるとする．その処理性能は，形状尺度 $k = 5$，尺度分母 $\Theta = 2/5$，期待値 2 のガンマ分布に従う．

悪人ノード

悪人ノードは，悪人の存在確率に応じて与えられる．また，エクスターナルグリッドの管理者は，悪人ノードと悪人ノードではないノードを区別できない．

悪人ノードの振る舞い

全ての悪人ノードは，正しい実行結果を返さない．かつ，ブロック内の各悪人ノード間で同種の実行結果を返す．

試行回数

同一入力における試行を 1000 回行う．

6. 評価結果と考察

5 節で述べた評価条件に基づいて，シミュレーションを行った結果を用いて評価を行った．本節では，その結果の提示と考察を行う．以降の図中のでは，プログラム分割数は 100，多重度は 20 である．

6.1 エクスターナルグリッドの安全性

悪人の存在確率とエクスターナルグリッドの総利用ノード数の平均の関係を，各高速化手法間で比較したものが，図 7 である．閾値暫定法と閾値網羅法の閾値は，共に 5 である．

悪人の存在確率が 0.2 において，閾値暫定法を利用したエクスターナルグリッドが利用したノード数は約 2000 である．閾値網羅法を利用した場合のノード数は約 2500 である．また，網羅法を利用した場合は，利用ノード数が 6300 となっている．

閾値網羅法と網羅法を比較した際には，悪人の存在確率が 0.2 において約 3 倍ほどノード数に違いが出る．また，

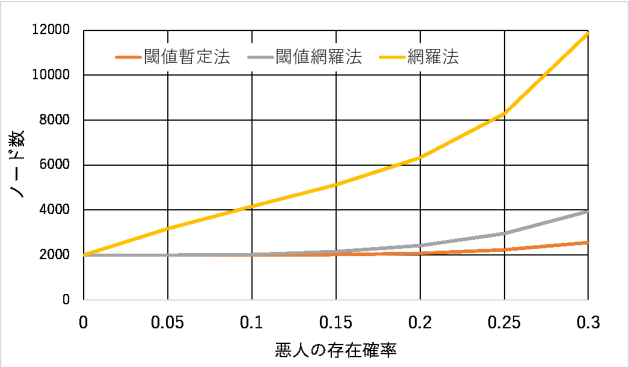

図 7　悪人の存在確率と総利用ノード数

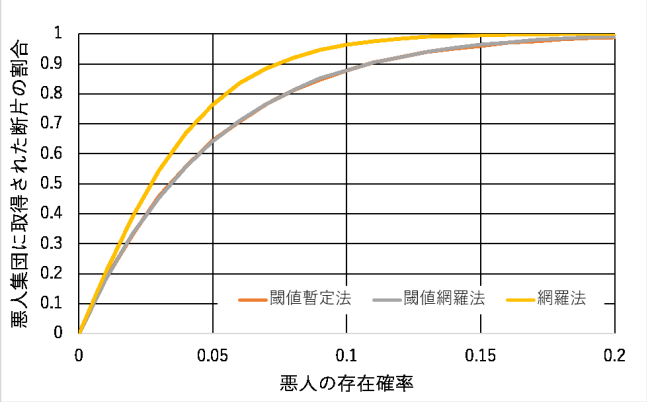

図 8　悪人の存在確率と悪人集団に取得されたプログラム断片の割合

シミュレーションの結果全体を見た場合に，閾値網羅法と網羅法のそれぞれのノード数のグラフ間には大きさがあるといえる．

一方で，閾値網羅法と閾値暫定法を比較した際には，大な差がないことがわかる．悪人の存在確率が 0.2 においても，閾値暫定法のノード数に対して，閾値網羅法のノード数は約 1.25 倍程度である．

以上の点から，エクスターナルグリッドの利用ノード数に関して，提案手法である閾値網羅法は，閾値暫定法と同程度のノード数を利用することがわかる．また，その利用ノード数も，網羅法と比較して少数で済むといえる．

6.2　悪人集団による不正解析のリスク

悪人の存在確率と悪人集団に取得されたプログラム断片の割合の関係を，各高速化手法間で比較したものが，図 8 である．閾値暫定法と閾値網羅法の閾値は，共に 5 である．図 8 全体を通して，閾値網羅法と閾値暫定法の各手法間の，悪人集団に取得された断片の割合はほぼ同様の結果を示している．

このような結果となった原因の一つは，悪人の振る舞いであると考えられる．今回の悪人の振る舞いは，1 つのブロックに所属する各悪人は，全員誤った結果を出し，かつ，互いに同じ内容を出す，というものである．この場合，閾値網羅法においては，1 つのブロックから，最大で 2 つまでしか処理の分岐が発生しない．かつ，閾値暫定法では，1 つのブロック内で先行処理は 1 つしか発生しない．更に，閾値網羅法と閾値暫定法では，先行処理のための閾値が同値であるため，図 8 のように似た傾向になったと考えられる．

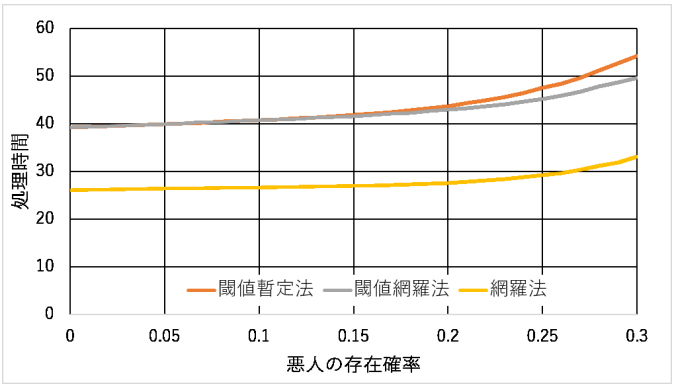

図 9　悪人の存在確率と処理時間

網羅法と閾値を用いる高速化手法の間で，悪人集団に取得される断片の割合の増加率が異なるのは，閾値によって，より信頼性の高い実行結果が先行処理に利用されれているためである．

6.3　エクスターナルグリッドの処理時間

6.3.1　処理時間に対する悪人の存在確率の影響

悪人の存在確率と処理時間の平均の関係を，各高速化手法間で比較したものが，図 9 である．閾値暫定法と閾値網羅法の閾値は，共に 5 である．

図 9 においても，高速化手法に閾値を用いるか否かで，結果がわかれている．

閾値網羅法と閾値暫定法に関して，悪人の存在確率が 0.2 程度になるまで，殆ど差が開いていない．悪人の存在確率が 0.2 の場合，閾値網羅法の処理時間は，閾値暫定法の処理時間に対して 2%程度の短さである．存在確率が 0.3 でも，閾値網羅法の処理時間は，閾値暫定法の処理時間に対する差は 10%程度である．

以上のような結果の原因は，6.2 節でも述べた原因と同様に，悪人の振る舞いによるところが大きいと考えられる．その結果，エクスターナルグリッドの先行処理の進め方が類似し，処理時間に関しても傾向が似通ったといえる．

悪人が存在しない段階から，閾値を用いた高速化手法より網羅法の処理時間が短くなっているのは，閾値による先行処理の遅れが原因であるといえる．

6.3.2　処理時間に対する閾値の影響

図 10 は，閾値網羅法と閾値暫定法における閾値と処理時間の関係を示している．閾値は，多重度の過半数より小さい値である必要がある．多重度は 20 であるので，閾値は 1 から 10 の範囲で変化する．

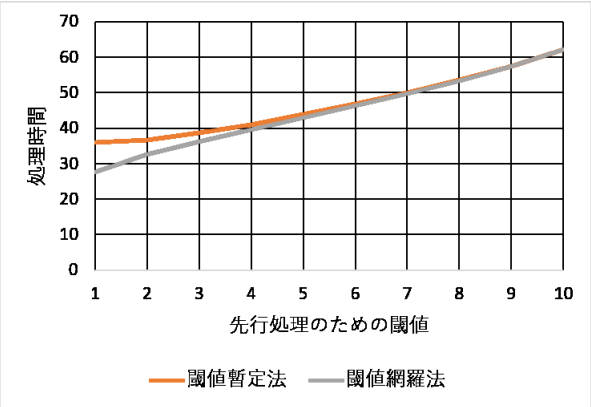

図 10 網羅法のブロック生成過程と処理時間

図 10 では，閾値が大きくなるにつれて，各手法の処理時間の差が小さくなっていくことがわかる．具体的には，閾値 2 の場合，閾値暫定法の処理時間は 36.6 であり，閾値網羅法の処理時間は 32.5 である．この段階で，閾値網羅法の処理時間は，閾値暫定法の処理時間に対して 12%程小さい．一方，閾値 10 においては，両手法ともに処理時間は 62.0 台となっている．

以上のことから，閾値が多重度の過半数に対してより小さい程，閾値網羅法は閾値暫定法よりも処理性能が高いことがわかる．

7. まとめ

本稿では，先行研究で提案されていたエクスターナルグリッドの高速化手法の一つである網羅法の問題を解決する，閾値網羅法を提案した．また，先行研究で提案されいる高速化手法である閾値暫定法と網羅法を提案手法を比較して，提案手法の特徴とエクスターナルグリッドに与える影響について，評価を行った．

利用ノード数に関して，提案手法は網羅法と比較して，ノード数をより抑えることができることが確認できた．また，提案手法と閾値暫定法を比較した際，利用ノード数は類似した傾向を示した．以上の点から，利用ノード数に関して，提案手法は網羅法より利用数を抑えることができ，閾値暫定法と同程度のノード数を利用することがわかった．

悪人集団に取得されたプログラム断片の割合について，利用ノード数と同じく，提案手法は閾値暫定法と同様の傾向を示した．更に，提案手法は，閾値暫定法と比較して，悪人に取得される断片の増加量が緩やかであることもわかった．

利用ノード数と悪人集団に取得された断片の割合から，提案手法は網羅法よりも不正行為によるリスクが低減できることがいえる．

エクスターナルグリッドの処理性能に関しては，エクスターナルグリッドに依頼された処理が終了するまでの時間を求めて評価した．その際，処理時間と，悪人の存在確率との関係または閾値との関係のそれぞれについて値を示した．

悪人の存在確率を変化させ提案手法と網羅法を比較した際，処理時間は網羅法の方が明らかに短いことがわかった．具体的には，悪人の存在確率 0.2 の場合，網羅法の処理時間は提案手法の 64%程の長さであった．また，処理時間に関して，提案手法と閾値暫定法の間では，殆ど差が開いていないことがわかった．

次に，閾値の違いによって処理時間に影響がでることがわかった．得られた結果より，閾値が小さい場合，提案手法の方が閾値暫定法よりも処理時間が短いことがわかった．しかし，閾値が多重度の過半数に近づくにつれて，提案手法と閾値暫定法の処理時間の差は小さくなり，収束する．

以上の，悪人による不正行為のリスクや処理性能に関して，閾値網羅法の特徴をまとめる．悪人による不正行為のリスクに関しては，閾値暫定法とは同程度で，網羅法よりも低いといえる．処理性能に関しては，網羅法と比較すると，閾値網羅法を取ることで低くなる．ただし，閾値を極力小さくすることで，処理時間の短縮を図ることができる．

謝辞

本研究は JSPS 科研費 JP26330105 の助成を受けたものです．

参考文献

[1] University of California: BOINC, http://boinc.berkeley.edu/（参照 2017-05-07）.
[2] University of California: SETI@home, http://setiathome.berkeley.edu/（参照 2017-05-07）.
[3] 平田智紀，稲元勉，樋上喜信，小林真也：セキュアプロセッシングにおけるファイル分散配置による通信負荷改善の効果に関する研究，マルチメディア，分散協調とモバイルシンポジウム 2014 年論文集，Vol. 2014，pp. 1806-1817(2014).
[4] 廣瀬吉隆，稲元勉，樋上喜信，小林真也：セキュアプロセッシングにおける先行処理による処理時間改善に対する定量的評価，情報科学技術フォーラム講演論文集，Vol. 14，No. 4，pp. 241-242(2015).
[5] 山口晃右，稲元勉，樋上喜信，小林真也：セキュアプロセッシングにおけるファイル分散配置による通信負荷改善の効果に関する研究，マルチメディア，分散協調とモバイルシンポジウム 2016 年論文集，Vol. 2016，pp. 344-351(2016).