社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ARC－172 (47)
2007－HPC－109 (47)
2007／3／2

ボランティアコンピューティングにおける信頼度評価に基づくジョブスケジューリング

渡邊 寛 福士 将 堀口 進
東北大学 大学院情報科学研究科

概 要

本稿では，誤った計算結果を返す妨害者が存在するボランティアコンピューティング (VC) 環境において，効率よく妨害者対策を行うための，信頼度評価に基づくジョブスケジューリング手法を提案する．従来，VC 環境内のワーカやジョブに対して信頼度を導入して妨害者対策を行う手法が提案されているが，ジョブの実行順については十分な検討がなされていない．本手法のアイデアは，ジョブに対して見込み信頼度を導入して，全ジョブが終了するまでの時間を小さくするようなジョブスケジューリングを行うことである．シミュレーションにより，従来用いられたラウンドロビン手法と比較して，実行時間が 7%程度改善されることが分かった．

Credibility-based job-scheduling for volunteer computing systems

Kan Watanabe, Masaru Fukushi, Susumu Horiguchi
Graduate School of Information Sciences, TOHOKU University

Abstract

This paper presents a credibility-based job-scheduling method on volunteer computing (VC) systems with malicious participants who submit erroneous results. The key idea is to introduce an expected credibility for jobs in order to select proper jobs and reduce the total execution time of sabotage-tolerant computing. This is a feature that no previous sabotage-tolerance methods possessed. It is shown by performance evaluation that our proposed method improves the execution time of sabotage-tolerant computing by 7%.

1. はじめに

ボランティアコンピューティング（VC）は，多数のボランティアによる計算資源の提供に基づく，大規模並列計算環境の実現方法である．VC では，ボランティアにより提供される，ネットワークに接続された計算資源の遊休 CPU サイクルを利用して，高性能な並列計算環境を安価に構築することができる．VC が世界規模で運用されている計算プロジェクトの例として，SETI@home[1] や distributed.net[2] などが挙げられる．

VC では，ボランティアにより提供される各計算資源（ワーカ）が，小規模に分割された計算（ジョブ）をそれぞれ独立に計算するため，参加者数が多いほど VC 環境全体の計算性能は高くなる．しかし，ボランティアの参加に制限を設けない場合は，誤った計算結果を故意に返すような悪意ある者（妨害者）がワーカの中に混じることがある．妨害者の数が多い場合は計算結果の信頼性が著しく低下するため，何らかの妨害者対策を行う必要がある．

妨害者対策を行うにあたっては，VC 環境の計算性能をなるべく低下させないような手法を採用するべきである．妨害者対策手法として，VC ミドルウェア BOINC[3),4)] で用いられている m-first-voting がある．これは，m 個の同一の計算結果が集まるまで，同じジョブを複数のワーカに計算させて，計算結果に冗長性を持たせる手法である．この手法は比較的簡単ではあるが，ジョブの終了に必要な冗長度を一律で m とするために効率が悪く，VC 環境の性能低下が大きい．VC ミドルウェア Bayanihan[5)] では，Luis によって，信頼度評価に基づく妨害者対策手法[6)] が提案されている．Luis の手法では，VC 環境内のワーカやジョブに対して信頼度という数値指標を導入し，信頼度をもとに各ジョブの終了に必要な冗長度を動的に算出することで，冗長なジョブをなるべく実行しないようにしている．

しかし，Luis の手法では，次にどのジョブを実行するかを決めるジョブスケジューリングに関しては十分に検討されていない．妨害者対策のためにジョブやワーカに信頼度を導入したことで，実行するジョブの順序により，全体の実行時間に違いが生じる点に注意しなければならない．そのため，適切なジョブスケ

ジューリングを行うことで，より効率よく妨害者対策を行い，VC 環境の性能低下を抑えることが可能になると思われる．

本稿では，信頼度評価に基づく妨害者対策を効率よく行うための，信頼度評価に基づくジョブスケジューリング手法を提案する．妨害者対策のために導入された信頼度を利用し，計算中のワーカの存在を考慮した，見込み信頼度という指標を新たに導入する．見込み信頼度を用いて，各ジョブの信頼度をできるだけ均一に上げるようなジョブスケジューリングを行うことにより，全てのジョブが終了するまでの実行時間を小さくする．

以下，2 節で本稿が扱う VC のモデルと妨害者対策について述べ，3 節で見込み信頼度を用いたジョブスケジューリング手法を提案する．4 節でシミュレーションによる評価を行い，5 節で本稿の結論と今後の課題を述べる．

2. VC のモデル

2.1 想定する VC 環境

VC の計算モデルとして，ワークプールを用いるマスタ/ワーカモデルを仮定する．これは，VC やグリッドコンピューティングなど，インターネットを用いた並列計算システムで広く用いられるものである．図 1 に，想定する VC 環境を示す．

- マスタ 1 台と，ボランティアにより提供されるワーカ W 台で，1 つの計算プロジェクトを実行する．
- 計算プロジェクトは N 個の独立した計算（ジョブ）よりなる．
- マスタはアイドル状態のワーカに対してジョブを 1 つ配布し，計算結果（リザルト）を回収する．この際，ワーカ間の通信は発生しない．
- 妨害者の存在率を f とする．ワーカ W 台のうち，$f \times W$ 台が妨害者である．
- 妨害者が誤ったリザルトを返す確率を，妨害率 s とする．s はマスタにとって未知の値である．

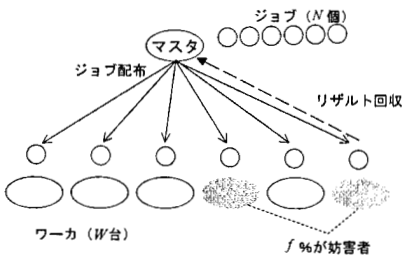


図 1 想定する VC 環境

ここで，ジョブの計算結果として確定したリザルトをアンサーと呼び，アンサーが得られた時点でジョブが終了するものとする．N 個の全てのジョブの終了をもって計算プロジェクトの終了とし，計算プロジェクトが終了するまでにかかった時間を実行時間 T とする．計算プロジェクトの終了時において，全ジョブのアンサーのうち，誤りであるアンサーの割合を計算プロジェクトの誤り率 ϵ とする．

妨害者対策を何も施さない場合，1 つのジョブに対して，1 台のワーカでリザルトを 1 つ生成するだけなので，リザルトの誤りが直接アンサーの誤りとなってしまう．各ワーカの性能やジョブの計算量が均一の場合，誤り率 ϵ は $\epsilon = \frac{N \times f \times s}{N} = f \times s$ となる．すなわち，妨害者の割合が多いほど誤り率 ϵ が大きくなり，計算結果の信頼性は著しく低下する．

2.2 妨害者対策

1 つのジョブに対して，複数のワーカからリザルトを集めて冗長性を持たせたり，VC 環境から妨害者を検出し排除するなどして，誤り率 ϵ を小さくすることを妨害者対策と呼ぶ．基本的な妨害者対策手法を表 1 に示す．これらの手法は，効率や計算結果の信頼性向上のために組み合わせて使用することが可能である．

Luis[6] は，表 1 に示すいくつかの手法を組み合わせた上で，さらに信頼度評価に基づいて，効率よく妨害者対策を行う手法を提案している．この手法では，ワーカやリザルトなどの VC 環境内の各要素に対して，信頼度という数値指標を導入する．

信頼度の導入により，あるジョブに対していくつのリザルトを集めれば十分高い信頼性を得られるかという冗長化問題に対して，ジョブの信頼度が閾値 θ に達するまで，という基準を設け，冗長なジョブの実行回数を抑制している．また，閾値 θ は任意に設定することが可能であるため，計算プロジェクトの誤り率 ϵ の期待値を，任意の許容誤り率 ϵ_{acc} 以下に保証することが可能となる．

表 1 基本的な妨害者対策手法

majority-voting	1 つのジョブに対して複数個のリザルトを集め，多数決を行う
m-first-voting	m 個の同一なリザルトが得られるまでジョブを配布し，リザルトを集める
spot-checking	予め正しい答えが分かっているジョブを計算させて，対象ワーカが妨害者かどうかチェックを行う
blacklisting	妨害者と判断されたワーカを隔離し，以降ジョブを割り当てない
back-tracking	妨害者と判断されたワーカがそれまでに生成したリザルトを無効化する

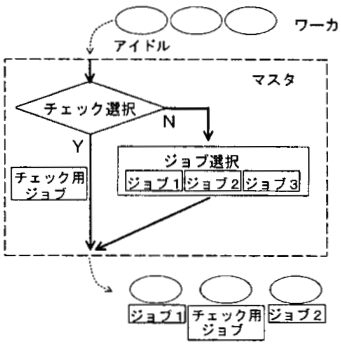


図 2　チェック選択とジョブ選択によるジョブスケジューリング

2.3　ジョブスケジューリング

妨害者対策を行う場合，一般に，1 つのジョブに対して複数のリザルトを集めて冗長性を持たせたり，spot-checking でチェック用のジョブを実行するため，全ジョブの終了に要する実行時間 T は，妨害者対策を行わない場合と比較して大きくなる．そのため，計算結果の信頼性を保証しつつ，実行時間 T を小さくすることが求められる．

本稿では，信頼度評価に基づく妨害者対策を行った場合のジョブスケジューリング問題を，次のように定義する．与えられた許容誤り率 ϵ_{acc} に対して，$\epsilon \leq \epsilon_{acc}$ を満たしつつ，実行時間 T をできるだけ小さくするように，アイドル状態のワーカに対して，割り当てるジョブを決定する問題．

一般に，アイドル状態のワーカに対しては，図 2 に示すように，チェック選択とジョブ選択の 2 つのプロセスによりジョブスケジューリングが行われる．チェック選択は，ワーカに対して spot-checking を行うかどうかを決定するプロセスである．spot-checking を行う場合には，ワーカにチェック用のジョブが配布されるため，計算プロジェクトの計算は進まない．チェック選択を行う基本的手法としては，一定確率 q で spot-checking を行い，$1-q$ で spot-checking を行わないランダム手法がある．

ジョブ選択は，計算プロジェクトの計算を進めるために，ワーカにどのジョブを割り当てるかを決定するプロセスである．ジョブ選択を行う基本的手法としては，乱数によってジョブを選択するランダム手法と，ジョブに付けられた番号順に選択するラウンドロビン手法がある．

3.　信頼度評価に基づくジョブスケジューリング

3.1　手法の概要

信頼度評価に基づく妨害者対策を効率よく行うための，信頼度評価に基づくジョブスケジューリング手法を提案する．本稿では，従来手法と同様に spot-checking, blacklisting, back-tracking を用いて，信頼度に基づく妨害者対策を行う．ジョブスケジューリングにおけるチェック選択では，チェック選択確率 q で spot-checking を行うランダム手法を用いるものとする．ジョブ選択では，次に実行するジョブを適切に選択するための指標として，見込み信頼度を新たに導入し，これを用いて実行時間 T を小さくするスケジューリング手法を提案する．

3.2　見込み信頼度

3.2.1　ジョブの信頼度

信頼度評価に基づく妨害者対策を行う際の，信頼度の計算方法を説明する[6]．blacklisting を行った場合，ワーカ w の信頼度 $C_W(w)$ は，spot-checking に通過した回数 k を用いて，式 (1) で表される．

$$C_W(w) = \begin{cases} 1 - \frac{f}{(1-f) \times ke} & \text{if } k \neq 0 \\ 1 - f & \text{if } k = 0 \end{cases} \quad (1)$$

ただし e は自然対数の底である．

リザルト r の信頼度 $C_R(r)$ は，そのリザルトを生成したワーカ w の信頼度 $C_W(w)$ と等しいものとする．

$$C_R(r) = C_W(w) \quad (2)$$

ジョブ j の信頼度 $C_J(j)$ は，j に対するリザルトが 1 個だけの場合は，そのリザルト r の信頼度と等しいものとする．

$$C_J(j) = C_R(r) \quad (3)$$

ジョブ j に対するリザルトが複数個あった場合は，値の等しい同一のリザルトごとにグループ分けを行い，各グループ（リザルトグループと呼ぶ）の信頼度からジョブの信頼度を算出する．ジョブ j のリザルトが g 個のリザルトグループ $G_{(j,1)}, G_{(j,2)}, \ldots, G_{(j,g)}$ に分けられたとすると，リザルトグループ $G_{(j,a)}$ の信頼度 $C_G(G_{(j,a)})$ は，式 (4)-(6) で計算される．

$$C_G(G_{(j,a)}) = P_T(G_{(j,a)}) \prod_{i \neq a} P_F(G_{(j,i)}) \Big/ \left(\prod_{i=1}^{g} P_F(G_{(j,i)}) + \sum_{n=1}^{g} P_T(G_{(j,n)}) \prod_{i \neq n} P_F(G_{(j,i)}) \right) \quad (4)$$

$$P_T(G_{(j,a)}) = \prod_{i=1}^{m_a} C_R(r_{ai}) \quad (5)$$

$$P_F(G_{(j,a)}) = \prod_{i=1}^{m_a} (1 - C_R(r_{ai})) \quad (6)$$

ただし，リザルトグループ $G_{(j,a)}$ は m_a 個のリザルト $r_{a1}, r_{a2}, \ldots, r_{a_{ma}}$ からなるものとする．このとき，ジョブ j の信頼度 $C_J(j)$ は，リザルトグループ $G_{(j,1)}, G_{(j,2)}, \ldots, G_{(j,g)}$ の中で最大の信頼度を持つグループである，$G_{(j,x)}$ の信頼度と定義される．

$$C_J(j) = C_G(G_{(j,x)}) = \max_{1 \le a \le g} C_G(G_{(j,a)}) \quad (7)$$

ジョブの信頼度が閾値 $\theta = 1 - \epsilon_{acc}$ を超えた場合，$G_{(j,x)}$ のリザルトをジョブのアンサーとし，ジョブを終了する．

3.2.2 ジョブの見込み信頼度

ジョブ j の信頼度は，図 3 に示すように，ワーカでの計算が終了し，回収したリザルトのみから計算される．その際，ジョブ j を現在計算中のワーカの存在は考慮されていない．ジョブを計算中のワーカからはリザルトが回収できると期待されるので，ジョブ選択を行う際に，計算中のワーカが存在するジョブと存在しないジョブの間で，なんらかの差別化を行うべきである．

各ジョブに対して，計算中のワーカの存在を考慮した，見込み信頼度という指標を新しく定義する．ジョブ j の見込み信頼度 $EC_J(j)$ は，ジョブ j を計算中の全てのワーカから得られるリザルトが，現在最も信頼度の高いリザルトグループ $G_{(j,x)}$ に属すると仮定した時の，ジョブの信頼度と定義する．

ジョブ j を現在 d 台のワーカで計算中であり，すでに g 個のリザルトグループが形成されているとする．このうち最大の信頼度を持つリザルトグループを $G_{(j,x)}$ とすると，ジョブ j の見込み信頼度 $EC_J(j)$ は，式 (8)-(10) で計算される．

$$EC_J(j) = \max_{1 \le a \le g} C'_G(G_{(j,a)}) \quad (8)$$

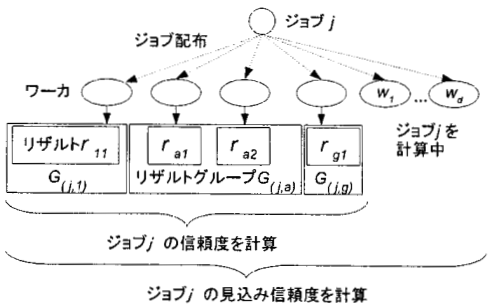


図 3 信頼度と見込み信頼度の違い

$$P'_T(G_{(j,a)}) = \quad (9)$$

$$\begin{cases} P_T(G_{(j,a)}) & \text{if } a \neq x \\ P_T(G_{(j,a)}) \times \prod_{i=1}^{d} C_W(w_i) & \text{if } a = x \end{cases}$$

$$P'_F(G_{(j,a)}) = \quad (10)$$

$$\begin{cases} P_F(G_{(j,a)}) & \text{if } a \neq x \\ P_F(G_{(j,a)}) \times \prod_{i=1}^{d} (1 - C_W(w_i)) & \text{if } a = x \end{cases}$$

ただし，$C'_G(G_{(j,a)})$ は，式 (4) における P_T, P_F の代わりに，式 (9)，(10) で表される P'_T, P'_F を用いて計算されるものとする．

また，ジョブ j のリザルトがまだ 1 つも生成されておらず，計算中のワーカのみしか存在しない場合は，$EC_J(j)$ は式 (11) で計算される．

$$EC_J(j) = \frac{\prod_{i=1}^{d} C_W(w_i)}{\prod_{i=1}^{d} C_W(w_i) + \prod_{i=1}^{d} (1 - C_W(w_i))} \quad (11)$$

3.3 見込み信頼度に基づくジョブ選択手法

見込み信頼度を用いることで，ワーカにジョブを割り当てた時点で，そのワーカがリザルトを返したときに信頼度に及ぼす影響を考慮することが可能となる．

例えば，単純に，各ジョブの現在の信頼度を用いてジョブスケジューリングを行う場合，あるワーカにあるジョブを割り当てても，ジョブの信頼度はリザルトが回収されるまで変化しないため，他のワーカにも同じジョブを割り当ててしまう可能性がある．すなわち，割り当てるべきワーカの台数を決定できないという問題がある．これに対し，見込み信頼度を用いる場合，ワーカにジョブを割り当てた時点で見込み信頼度は更新されるため，このような問題は生じなくなる．

次に，見込み信頼度を用いる場合に，どのジョブを優先的に実行するべきかを考える．ここで，spot-checking がジョブの信頼度に与える影響を考える．各ワーカが spot-checking を通過する回数は，時間が経過するにつれて単純に増加する．そのため，各ワーカの信頼度は時間とともに増加するので，ジョブの信頼度も時間とともに増加する傾向にある．ただし，ワーカの信頼度が増加しても，終了したジョブの信頼度には影響しない．このことから，1 つのジョブをできるだけ早く終わらせるような戦略よりは，各ジョブの信頼度をできるだけ均一にして，ジョブが終了するタイミングを揃えるような戦略のほうがよいと考えられる．

よって，本提案手法では，ジョブ選択の戦略として，ジョブの見込み信頼度が最も低いジョブを選択するようにジョブスケジューリングを行う．

4. シミュレーション

4.1 シミュレーションの仮定

提案手法の性能を評価するために，VC 環境におけるジョブスケジューリングのシミュレーションを行った．ジョブスケジューリング手法として，ジョブ選択で，提案手法，ラウンドロビン手法，ランダム手法を行う場合の，実行時間と誤り率を，シミュレーション 1000 回の平均値で評価した．

各手法は，チェック選択では，チェック選択確率 q で spot-checking を行い，妨害者が検出された場合は，blacklisting によりそのワーカを隔離し，back-tracking によりリザルトを無効化するものとした．

VC 環境については，従来法での評価と同様に，表 2 に示すパラメータを仮定し，また以下の仮定をおいた．

- 各妨害者は，他の妨害者とは独立に妨害を行い，妨害を行う際はランダムな値をリザルトとする．
- 各ジョブの計算量およびワーカの性能は均一で常に一定とし，それぞれのジョブ計算に単位時間 1 がかかるものとする．また，妨害者が誤ったリザルトを生成する場合にも同様に単位時間 1 がかかるものとする．

4.2 シミュレーションの結果

4.2.1 f に対する実行時間の比較

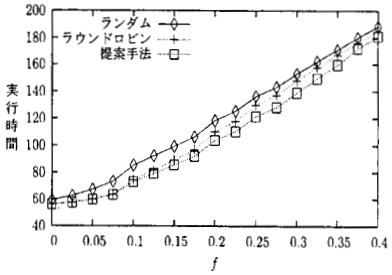
図 4 に，$s = 0.5$ とした場合の，f に対する実行時間を示す．f が増加するに従い，誤ったリザルトが返される確率が高くなるため，どの手法においても実行時間が増加する傾向がみられる．ランダム手法に比べ，提案手法とラウンドロビン手法は実行時間が小さい．

提案手法とラウンドロビン手法を比較すると，f が小さい場合は ϵ_{acc} が 0.01，0.0001 のどちらの場合においても，実行時間に差はない．しかし，ある程度 f が大きくなると，提案手法の方が実行時間が小さくなっていることが分かる．例えば，$\epsilon_{acc} = 0.01$ の場合は $f = 0.2$ において，ラウンドロビン手法を用いた場合の実行時間 110 に対して提案手法では 103 と，約 7%の高速化が見られた．

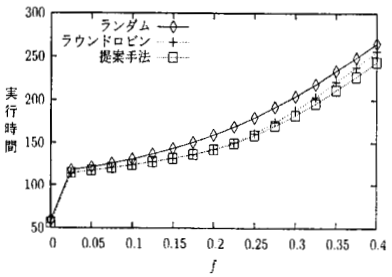
この理由は次のように考えられる．両手法では，始めは各ジョブに対して同じ個数のリザルトが生成されが，spot-checking により妨害者 X が検出されると，

表 2 パラメータ

ジョブ数 N	1000
ワーカ数 W	200
チェック選択確率 q	0.1
許容誤り率 ϵ_{acc}	0.01 , 0.0001
妨害者の存在率 f	0.00 – 0.40
妨害率 s	0.00 – 1.00



(a) $\epsilon_{acc} = 0.01$ の場合



(b) $\epsilon_{acc} = 0.0001$ の場合

図 4 $s = 0.5$ の場合の f に対する実行時間

back-tracking により X の生成したリザルトは無効化されて，各ジョブのリザルト数に偏りが発生する．この際，ラウンドロビン手法ではジョブの順番のみでジョブ選択を行うために，この偏りは解消されない．しかし，提案手法では，見込み信頼度の計算によってリザルト数が少なくなったジョブが分かり，ジョブ選択はリザルト数の偏りを解消するよう行われるため，実行時間が小さくなる．しかし，f が小さく，妨害者があまり存在しない場合は，back-tracking によるリザルト数の偏りが発生しにくいため，両手法の実行時間に大きな差が出ない．

4.2.2 f に対する誤り率の比較

図 5 に，$s = 0.5$，$\epsilon_{acc} = 0.01$ とした場合の，f に対する誤り率を示す．手法によらず，どのような f に対しても，誤り率 $\epsilon \leq \epsilon_{acc}$ が満たされていることが分かる．特に，ラウンドロビン手法と提案手法では誤り率はほぼ同じとなった．

4.2.3 s に対する実行時間の比較

図 6 に，$f = 0.3$ とした場合の，s に対する実行時間を示す．ϵ_{acc} が 0.01，0.0001 のどちらの場合においても，ある程度 s が大きい場合は，ラウンドロビン手法より提案手法の方が実行時間が小さくなっていることが分かる．例えば $\epsilon_{acc} = 0.01$ の場合は $s = 0.5$ において，ラウンドロビン手法を用いた場合の実行時間 148 に対して提案手法では 139 と，約 6%の高速化

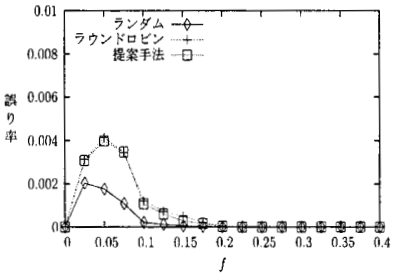


図 5　$s = 0.5, \epsilon_{acc} = 0.01$ の場合の f に対する誤り率

が見られた．

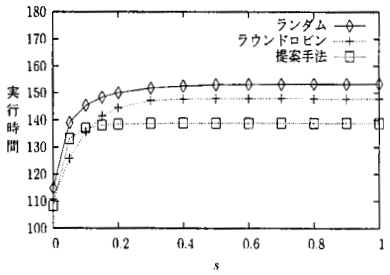
しかし，図 6(a) に示されるように，s が非常に小さい場合，ラウンドロビン手法の方が提案手法より実行時間が小さい場合があった．

この理由は次のように考えられる．まず，妨害者は $1-s$ の確率で spot-checking に通過し，信頼度を増加させるため，s が小さいときには，信頼度の高い妨害者が存在する場合がある．ここで，信頼度の高い妨害者が，あるジョブ j に対して誤ったリザルトを生成すると，このリザルトの影響で，正解のリザルトグループの信頼度は非常に上がりにくくなる．誤ったリザルトは，それを生成した妨害者が spot-checking に通過しなかった場合に back-tracking により無効化されるが，s が小さい場合はこの無効化も起こりにくい．このような場合，ジョブ j の見込み信頼度も増加しにくいため，提案手法では多数のワーカがジョブ j の計算を割り当てられてしまい，他のジョブがあまり進まなくなってしまう．

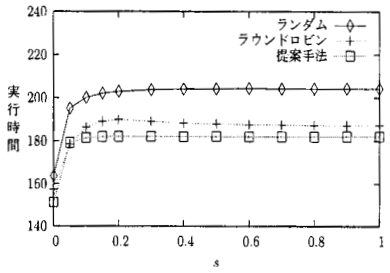
この問題を回避するには，あるジョブを同時に計算するワーカ数を一定数以下に制限する方法や，このような問題を発生させているリザルトを発見し，それを生成したワーカには spot-checking を優先して行う方法などが考えられる．

5. ま と め

本稿では，妨害者が存在する VC 環境において，信頼度評価に基づく妨害者対策を効率よく行う，信頼度評価に基づくジョブスケジューリング手法を提案した．提案手法は，見込み信頼度に基づいて，各ジョブの信頼度をできるだけ均一に上げるようなジョブスケジューリングを行う．シミュレーションにより，ほとんどの場合において，提案手法によって実行時間が小さくなることを示した．提案手法による計算プロジェクトの誤り率 ϵ は，従来のラウンドロビン手法の場合とほぼ同じで，許容誤り率 ϵ_{acc} 以下を保証することを確認した．



(a) $\epsilon_{acc} = 0.01$ の場合



(b) $\epsilon_{acc} = 0.0001$ の場合

図 6　$f = 0.3$ の場合の s に対する実行時間

今後の課題として，ジョブの重みとワーカの性能がそれぞれ均一でない場合や，ワーカが VC 環境から離脱するような場合を扱うことが挙げられる．VC 環境では，計算中に突然電源が切られるなどして，ワーカが VC 環境から離脱してしまい，リザルトを回収することができない場合があるので，タイムアウト τ を設定する必要がある．見込み信頼度を，時間的に減衰するように計算方法を拡張し，τ の決定に用いることで，適切な τ を動的に設定することができると考えられる．

謝辞　本研究の一部は総務省戦略的情報通信研究開発推進制度 SCOPE 特定領域重点型研究開発 (061102002) 助成によって行われた．関係各位に感謝する．

参 考 文 献

1) http://setiathome.ssl.berkeley.edu/
2) http://www.distributed.net/
3) http://boinc.berkeley.edu/
4) David P.Anderson, "BOINC: A System for Public-Resource Computing and Storage", 5th IEEE/ACM International Workshop on Grid Computing, pp.4-10, Nov. 2004.
5) http://www.cag.lcs.mit.edu/bayanihan/
6) Luis F. G. Sarmenta, "Sabotage-Tolerance Mechanisms for Volunteer Computing Systems", Future Generation Computer Systems, Vol. 18, Issue 4, pp.561-572, 2002.