様式Ｃ－１９

科学研究費助成事業（科学研究費補助金）研究成果報告書

平成２４年　５月３１日現在

機関番号：17501
研究種目：基盤研究（C）
研究期間：2009～2011
課題番号：21500280
研究課題名（和文）
ヘテロ並列分散計算環境による統計計算アルゴリズムの効率化に関する研究
研究課題名（英文）
EFFICIENT ALGORITHMS FOR STATISTICAL CALCULATION IN HETEROGENIC PARALLEL DISTRIBUTED COMPUTATIONAL ENVIRONMENT
研究代表者
越智　義道（OCHI YOSHIMICHI）
大分大学・工学部・教授
研究者番号：60185618

研究成果の概要（和文）：研究室内の多様なPCをLANで結合した形での並列分散計算環境で，統計計算アルゴリズムの並列化の検討を行った．まず，モンテカルロ積分を中心にタスク配分の基礎的な枠組みとして，事前分配法・ワークプール法での並列化効率とその改善について検討し，その後，ロジスティック回帰分析での計算を中心とした正確推論・ベイズ推論，樹木接近法での計算アルゴリズムの並列化のアプローチについてその適否について検討した．

研究成果の概要（英文）：Possibilities of parallelization for algorithms in statistical calculation are explored in a parallel distributed computational environment where various PCs in a laboratory are inter-connected via LAN. Efficiency and remedies of pre-distribution and work-pool approaches are examined as a basis of task distribution scheme in the context of Monte-Carlo integration. Then, appropriateness of their parallelism approaches is evaluated for calculation algorithms for statistical methods, such as, exact and Bayes inferences for logistic regression analysis and classification tree based approach.

交付決定額

（金額単位：円）

	直接経費	間接経費	合　計
2009年度	2,000,000	600,000	2,600,000
2010年度	700,000	210,000	910,000
2011年度	700,000	210,000	910,000
年度			
年度			
総　計	3,400,000	1,020,000	4,420,000

研究分野：総合領域
科研費の分科・細目：統計科学
キーワード：計算機統計，並列計算，アルゴリズム

１．研究開始当初の背景

計算機を支える技術の急速な進歩によりPCの処理能力は大幅に向上し，研究者が日常的に使用するPC単体でも複数のCPUを備えたものや32ビットのアドレスの範囲を越える物理メモリを搭載するものが登場するなど，その性能の向上は著しい．

このことは，統計科学における計算の側面を大きく変貌させ，従来ではその実行時間あるいは計算機のメモリの制約のため使用をためらわれた方法であっても，実用的な分析方法として考えられるようになってきてい

る．このように計算負荷の高い方法でも実用的なものと考えられることとなったため，統計的な手法開発の観点からは，そのような方法の詳細な特性評価を試みる目的のために，さらなる計算処理能力の向上への期待を生み出すこととなっている．

このようなニーズに応えるためにネットワークで接続された複数の計算機を利用して並列に処理を行わせることによってより高速な処理能力を得ることも考えられ，並列分散処理による統計計算の研究が統計科学でも注目されている．

２．研究の目的

本研究では，計算機が不均一な状態のもとでネットワーク接続された並列分散処理環境下において効率の高い統計計算上のアルゴリズムの開発と評価を目的とする．その際に，タスク処理時間の変動やタスク間依存性に配慮し，タスク分割の方式やそのタスク・スケジューリングの戦略について効率的な方策の提案を行うことを研究目的とした．

３．研究の方法

本研究は，不均一な計算機群からなる並列処理環境のもとでの統計計算の計算アルゴリズムの開発を目的とした．このために，

（１）統計的計算の観点から汎用手法と考えられるシミュレーション的技法における基本枠組みに関する検討

（２）離散データ解析における正確推測として，ロジスティック回帰モデルを基礎とする正確推測への適用

（３）ベイズ推測や EM アルゴリズムにおける確率や期待値計算等で必要とされるモンテカルロ積分や MCMC アルゴリズムへの適用

（４）樹木接近法アルゴリズムへの適用

について，タスク依存性や処理時間の特性に応じたそのタスク分割の方策と並列化アルゴリズム，タスク管理手法について検討した．

今回の並列計算環境としては，研究代表者の研究室の計算機群において，Linux を OS として採用し，並列計算用開発実行環境としては，Message Passing Interface(MPI)準拠の環境， OpenMPI(http://www.open-mpi.org/)を利用することとした．

４．研究成果

（１）シミュレーション的技法における基本的枠組みに関する検討

一般に，並列計算においては，どのように実行すべきタスクを並列化して考えるかが重要なポイントとなる．本研究では図-1 に示す２つの方法を基礎に基本枠組みを考えることとした．

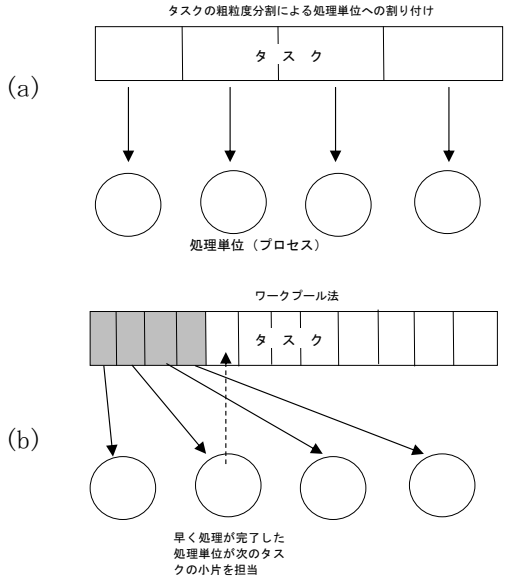

図-1 並列処理におけるタスク配分

図-1(a)は処理単位に対して大きくタスク分割し配置する方法であり，(b)はタスクを細かく分割し，処理の済んだ処理単位順に次のタスクを割り当てる方法である．ここでは(a)を事前分配法，(b)ワークプール法と呼ぶ事にする．ここで問題となるのは，計算機が均質な状況でない時，各計算機の処理速度を加味して適切なタスク配分を行う事である．このときワークプール法はその構成自身から，ヘテロな環境においてタスク配分を自動的に調節できることが考えられるが，事前分配法では，単純に CPU クロックでは実タスクの処理を計測できないことがあり，タスクの小区分を実際に実施して時間予測を行う事が求められる．

図-2 の様に矩形内に乱数により点を配置して，扇形内の点の計数値をもとに，円周率πを求める，モンテカルロ積分を考えた．10 億個の点を配置する場合の，事前予測法での処理時間計測小区分について検討したところ，10 万個までででは処理結果が安定しなかった．20 万個以降は実行時間が暫減し始め，500 万個(全体の 0.005%)を越えたところで小区分処理でのアイドル時間が問題となり処理時間が増加傾向に転じることが分かった．

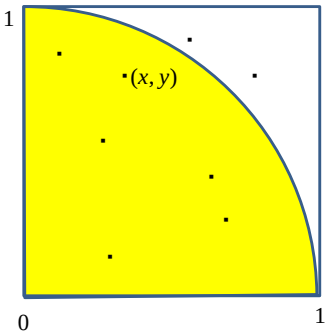

図-2 モンテカルロ積分

一方で，ワークプール法ではタスク区分を多くすると全体的なロードバランスは調整できるものの通信負荷が多くなり，タスク区

分を小さく取ると1区分の処理量が大きくなり最終段階でのアイドル時間が問題となる．そこで，最初は区分を大きく取り，後半になるにつれその区分を小さく取るようにタスク区分を調整することを検討した．その結果，均等にタスクを分配するよりは，そのタスク量を等比数列的に減少させる方が，計算結果を安定して減少させる効果を生むことが確認できた．ただし，その効果は 10%程度の処理時間の短縮であった．タスク分配に対して複雑な機構を導入するメリットについては，（5）で述べるような最近の CPU のアーキテクチャの変化への対応も併せて考慮することが必要である．

（2）ロジスティック回帰モデルでの正確推測に関する検討

対象個体 i (i=1,2,…,N)の 2 値反応を Y_i，その反応確率を p_i (=Pr(Y_i=1))，説明変数ベクトルを $\mathbf{x}_i$ とするとき，ロジスティック回帰モデルは

$$\log\frac{p_i}{1-p_i}=\mathbf{x}_i^T\boldsymbol{\beta}\quad(i=1,\ldots,N)$$

と書ける．ここで，$\boldsymbol{\beta}$ は説明変数に対応する回帰パラメータである．このときこのモデルの尤度は

$$L(\boldsymbol{\beta})=\frac{\exp(\mathbf{t}^{*T}\boldsymbol{\beta})}{\prod(1+\exp(\mathbf{x}_i\boldsymbol{\beta})}$$

と書ける．このとき $\mathbf{t}^*$は $\boldsymbol{\beta}$ の十分統計量

$$\mathbf{T}=\sum Y_i\mathbf{x}_i$$

の実現値である．

ここで，パラメータ $\boldsymbol{\beta}$ について $\boldsymbol{\beta}^T=(\boldsymbol{\beta}_1{}^T,\boldsymbol{\beta}_2{}^T)$ とし，$\boldsymbol{\beta}_1$ を局外パラメータ，$\boldsymbol{\beta}_2$ を目的パラメータとする．このとき正確推測として $\boldsymbol{\beta}_2$ に関する仮説検定 $H_0:\boldsymbol{\beta}_2=\boldsymbol{\beta}_2{}^*$ を考えると，条件無し推測では，$\boldsymbol{\beta}_2=\boldsymbol{\beta}_2{}^*$のもとで

$$P=\max_{\boldsymbol{\beta}_1}\sum_{\Pr(\mathbf{T}=\mathbf{t})\le\Pr(\mathbf{T}=\mathbf{t}^*)}\Pr(\mathbf{T}=\mathbf{t})$$

により P-値を求めることになる．ただし，ここで，

$$\Pr(\mathbf{T}=\mathbf{t})=\frac{c(\mathbf{t})\exp(\mathbf{t}^T\boldsymbol{\beta})}{\prod(1+\exp(\mathbf{x}_i\boldsymbol{\beta}))}$$

であり $c(\mathbf{t})$は十分統計量の実現値が $\mathbf{t}$ であるような反応 y_i からなる Nビット 2 値系列の個数である．

$\mathbf{T}_1$ について条件付けを行って条件付分布を考える場合には，

$$\Pr(\mathbf{T}_2=\mathbf{t}_2\mid\mathbf{T}_1=\mathbf{t}_1^*)=\frac{c(\mathbf{t}_1^*,\mathbf{t}_2)\exp(\mathbf{t}_2^T\boldsymbol{\beta}_2)}{\sum_{\mathbf{u}}c(\mathbf{t}_1^*,\mathbf{u})\exp(\mathbf{u}_2^T\boldsymbol{\beta}_2)}$$

として P–値を

$$P=\sum_{\mathbf{u}\in\left\{\mathbf{u}\mid\Pr(\mathbf{T}_2=\mathbf{u}\mid\mathbf{T}_1=\mathbf{t}_1^*)\le\Pr(\mathbf{T}_2=\mathbf{t}_2^*\mid\mathbf{T}_1=\mathbf{t}_1^*)\right\}}\Pr(\mathbf{T}_2=\mathbf{u}\mid\mathbf{T}_1=\mathbf{t}_1^*)$$

によって求めればよい．

これらの計算処理についての並列化については，条件付き分布の場合は，基本的に，$(\mathbf{t}_1{}^*,\mathbf{t}_2)$のパターンとそれぞれに対応する 2 値系列の個数 $c(\mathbf{t}_1{}^*,\mathbf{t}_2)$の数え上げの処理を行うことが，その計算負荷の大半である．この計算については MSA (Hirji, Mehta, and Patel, 1987)と呼ばれるアルゴリズムが存在するが，このアルゴリズムは逐次計算に基づく再帰計算を基礎とするために，タスクの粗な分解が困難である．ベクトル演算に関わる細部での並列化は可能であるが，ベクトルの次元が高くない場合，MPI を基礎とする並列処理では通信負荷が全体の処理時間に悪影響を及ぼす結果となることが確認された．後述（5）の共有メモリ型の並列処理では，この観点での改善が期待される．

一方で 2^N個の 2 値系列の全数列挙に基づく計算では，その並列化のためのタスク分解は比較的容易である．条件なし推測では，この計算を基礎とする．また，パラメータ $\boldsymbol{\beta}_1$ での最大化の処理では，離散データの確率計算の特性から，考えるべき和の結果はパラメータに対して非連続な関数となり，パラメータ $\boldsymbol{\beta}_1$ での網羅的な探索が要求される．この探索段階で並列化を考えることが可能である．条件なし正確推測法について，これらの観点について，並列化を行い，（1）で検討した事前計画法でのタスク分散のアプローチを導入することによって，5 台の計算機（5 プロセス）によって並列処理することによって，逐次計算の場合と比較して，ほぼ 1/4 まで処理時間を短縮することが可能であった．

（3）ベイズ推測における MCMC に関する検討

（2）と同様な設定のもとで，パラメータ $\boldsymbol{\beta}$ に関わる推測を，ベイズ推測の枠組みで考えることとした．この際に，$\boldsymbol{\beta}$ の事前分布としては，多変量正規分布を想定し，マルコフ連鎖によって事後分布をシミュレーションとして生成する MCMC 法の適用を考えた．

具体的な計算アルゴリズムとしては Metropolis-Hastings(MH)法を採用した．このとき連鎖はマルコフ性を伴って系列が生成されるために逐次計算の形態を取る．従って（2）の MSA の場合と同様，細粒子化したタスクレベルでの並列化は期待できるが，1 系列生成内での並列化は MPI には適さない．

ただし，MCMC の場合，事後分布からの実現値としての乱数生成がその目的であることから，複数の系列を生成し，それらをマージして利用することが可能である．本研究では

系列生成を複数プロセスで独立して行う形での並列化を考えた．また，各系列生成にあたっては独立して1本の連鎖としての系列を生成しなければならない．このためワークプールによるタスク分割法はなじまない．

並列処理の際に留意すべき点の一つは，各プロセスで使用する基礎乱数の生成方法であるが，これも各プロセス中に埋め込んで利用することとした．並列分散環境では並列処理のためにクロックの調整を行っており，機械が別であっても，デフォルトで使用すると各プロセスで同一の基礎乱数列を用いる可能性がある．このためシード等を調整して，同一乱数列とならない配慮が必要である．

また，MH によるマルコフ連鎖では，初期値から分布として安定して，定常状態に推移するまでに時間がかかる．通常これらの安定状態に移行するまでの系列はバーンインとして捨てられる．長い1本の系列であれば，多少のバーンインは問題でないが，複数のプロセスでこれを分割して生成し，プロセスごとの生成系列の短縮を狙っているので，この際のバーンインの設定には注意が必要である．つまり，バーンインとしては，各プロセスごとに1本の系列の場合とほぼ同様に必要であるので，バーンイン以降の系列を十分に確保するよう設計することが重要である．

5 パラメータもつロジスティック回帰モデルで，標本サイズ 100 のデータに対して HM 法を適用し，300 万個の MCMC ベクトル系列を生成する実験では，逐次計算で 992.2 秒かかる計算が，3 台の計算機を利用することによって 432 秒と約 2.3 倍の処理効率の向上を確認した．

（4）樹木接近法アルゴリズムへの適用に関する検討

判別木によるデータの分析に関するアルゴリズムについても検討を行った．木の分類にあたっては，2 分木の構成手順の繰り返しとなり，ある節点からの分岐はそれまでのデータ集合の結果に依存する事になるが，一旦分岐した後の木の構成は独立に作業を進めることができる．このため木の構成においては，ある程度の深さまで進行した段階で，各ノード以降の木の生成処理を並列に行う事が考えられる．ただし，今回検討材料としたデータ集合は 768 件・8 説明変数のデータであったため，この規模のデータの分析での並列化では，MPI を用いた場合の通信のオーバーヘッドの方が重たい状況であった．

従って，ここでは，Breiman ら(1999)の提唱するランダム・フォレスト法への適用について検討した．この方法では，データからのブートストラップ・サンプルに対して緩やかな基準のもとで木を構成し，この繰り返しで得られた木をもとにアンサンブル判定を行うことを基礎としている．この処理をブートストラップ・サンプル抽出から判別木構成までを基本処理タスクとして事前分配法で並列に実施することを考えた．生成する木の大きさを 400，800，1200，1600 とした時の結果は以下の表-1 の通りである．この表の時間は，誤判別を計測するために練成集合とテスト集合を取り変えて 100 回繰り返し実行した結果である．また並列環境では 9 台の計算機を用いているが，実際の木の構成を担当した計算機は 8 台である．この結果から，この例の場合の処理効率は約 5 倍であることが確認できる．

表-1 ランダム・フォレストでの
木の数と処理時間変化

木の数	単一マシン（分）	並列環境(分)
400	127.33	24.94
800	260.27	46.62
1200	377.99	67.63
1600	472.56	88.22

（5）スレッド並列に関わる検討

現在のアプリケーション開発環境としては，プロセス並列を前提として，スケーラビリティに優れたメッセージパッシングによる分散メモリ・分散処理を基軸にした環境の他に，単一計算機内で共有メモリ形でスレッドを基盤にした開発環境を利用可能である．本研究期間内に計算機の CPU のマルチコア化が急速に進展した．研究開始当初 2009 年時点では 2 コアを持つものがマルチコアとしては主流で，4 コアのものが高価ではあるがようやく入手可能になっている状況であった．ところが 2010 年に 6 コアを持つ CPU が Intel や AMD から市場に投入されるようになり，2011 年後半には 8 コアの CPU が出現することになった．Intel の CPU ではハイパースレッディングにより見かけ上実装コア数の倍のコアを持つように動作することが可能となっている．このことにより，単一計算機内での並列計算の可能性が大きく進むことになった．従って，本研究課題での分散計算環境についても，単一計算機内でのスレッド並列を基盤として並列計算を考える必要が出てきた．このために，スレッド並列に関わる検討も一部行った．

ここで言うスレッド並列では，共有メモリ形計算を基本とする．計算機内部では図-3 のように，プロセスを中心に計算処理が行われ，各プロセスは主メモリの一部をそのプロセス用に確保し，一つの処理体系として自律的に処理が行われる．並列処理が可能なシステムでは，このプロセスを複数起動することが可能である．プロセスを基盤とする並列処理では，プロセス間で情報の共有はメッセージパッシング等のプロセス間通信によって行

われる．

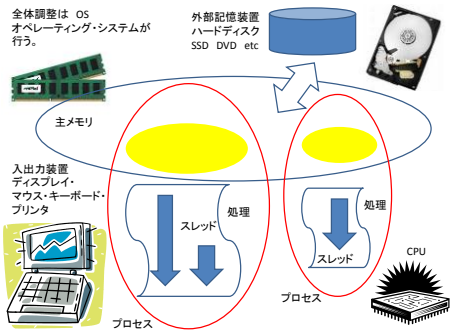

図-3 プロセスとスレッド

一方各プロセス内では，データの処理の流れはスレッドとして捉えられ，スレッド並列が可能なシステムでは，プロセス内で複数のスレッドを実行することが可能になる．この技術の実装として，ここでは OpenMP(http://openmp.org/wp/)を採用した．この実装では，各スレッド間の情報共有は共有メモリを通じて行われることになり，プロセス並列の場合のような通信に関わるオーバーヘッドを回避できる可能性がある．

そこで，（1）で検討した，円周率πを求めるためのモンテカルロ計算について OpenMP を用いて実装を行い処理時間の検討を行った．

表-2 OpenMP 乱数点 1 千万個による結果
-標準 C ライブラリの乱数生成器-

Intel Core i7 970 @3.20GHz 6 コア，12 スレッド		AMD PhenomII X6 1090T @3.2Ghz 6 コア，6 スレッド	
スレッド数	処理時間（秒）	スレッド数	処理時間（秒）
1	0.250840	1	0.331953
12	2.526189	6	13.85282
20	2.735719	12	4.605384

結果としては表-2 の様に単一スレッドのものより遅くなってしまった．

MPI についても，乱数点 1 千万個の結果では，OpenMP の場合と同様に，処理時間は長くなってしまっている．ところが，乱数点の生成を 1 億個とした場合には，プロセス数を増した方が早く結果を得ることが示された(表-3)．OpenMP の結果では，乱数の生成数を増してもこの逆転現象は見られなかった．つまり OpenMPI での結果では，プロセス間通信のためのオーバーヘッドがその処理に対して過負荷な状況であったと考えられる．

表-3 OpenMPI による結果

Intel Corei7 970 @3.20GHz 6 コア，12 スレッド		AMD PhenomII X6 1090T @3.2Ghz 6 コア，6 スレッド	
プロセス数	処理時間（秒）	プロセス数	処理時間（秒）
乱数点 1 千万個			
1	0.2507526	1	0.339060
10	1.071329	5	1.352831
乱数点 1 億個			
1	23.201364	1	32.310273
10	5.169170	5	8.256585

一般にメモリ共有型の並列処理の場合，共有変数に対する並列処理についてデータレースと呼ばれるアクセスタイミングの問題が生じることが知られている．このため，OpenMP を用いる際に，乱数をあらかじめ単一スレッドで生成しておき，そのデータの処理に特化した形で並列処理させたところ，1 千万個の乱数を用いた処理で，Intel CPU，12 スレッドで 0.2477 秒，AMD CPU で 0.3722 秒とそれぞれ単一スレッドで実施した結果，0.3064 秒，0.4293 秒よりも短くなった．このことから，通信負荷の問題を回避できる様子が確認でき，乱数生成ライブラリにその問題があることが分かった．

そこで標準の C ライブラリでの乱数生成器を GSL（GNU Scientific Library)ライブラリの乱数生成器に変えて，乱数生成の部分も含めて並列実行した結果が表-4 である．

表-4 OpenMP 乱数点 1 千万個による結果
-GSL ライブラリの乱数生成器-

Intel Core i7 970 @3.20GHz 6 コア，12 スレッド		AMD PhenomII X6 1090T @3.2Ghz 6 コア，6 スレッド	
スレッド数	処理時間（秒）	スレッド数	処理時間（秒）
1	0.271904	1	0.326749
6	0.050712	6	0.110705
12	0.048674	12	0.115328

並列処理の効果が適切に反映された結果を得ていることが確認できる．また，この結果は 1 千万個のものであるが，MPI の場合と比較して，通信負荷によるオーバーヘッドの問題も回避でき，大きく結果を改善できる可能性をスレッド並列が持つことが確認できる．また，Intel のハイパースレッディングによる効果は，今回のようなモンテカルロ積分の処理としては期待されるほどの効果は生んでおらず，モンテカルロ法の様な計算集約的な方法では，基本的にその物理的なコア数によって処理の効率化を考えるべきであることが示唆された．

（6）今後の課題

本研究の成果（1）～（5）で示したように，統計計算においても，用いるべきアルゴリズムと並列処理のテクニックがその結果

に大きく影響を与えることが確認できた．（５）の結果自体は単一計算機の結果であるが，並列分散環境にあって，単一計算機の処理効率を上げておくことは，全体的な効果に大きな影響を与える．プロセス並列と比較して，スレッド並列はプロセス間通信を持たないために，並列化の直接的な効果が期待できる．一方で，このことは情報共有を共有メモリに依存することを意味し，このためにスレッド並列の場合は，そのスケーラビリティに制約を生じることになる．

従って，複数の計算機を用いた並列計算の実装にあたっては，計算機間の処理としてはプロセス並列を基本としスケーラビリティを確保し，計算機内ではスレッド並列を用いて効率を高めるハイブリッド型の並列処理を今後検討する必要がある．そこでは本研究で基本としていた分散メモリ形計算から，共有メモリ形計算の観点を踏まえた各統計計算の処理の並列化の方策について再考することが必要である．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔学会発表〕（計 8 件）

①越智義道，並列計算環境下における統計計算アルゴリズムの適応，科研費研究集会「多変量解析の新展開(New developments in multivariate analysis)」，2012年1月19日，那覇市

②Kentaro Kuroishi，Yoshimichi Ochi，Properties and Effects of Inferences onHierarchical Generalized Linear Models in Clinical Trial Designs，Joint Meetingof the 2011 Taipei International Statistical Symposium and 7th Conference of the Asian Regional Section of the IASC，2011年12月17日，Taipei

③越智義道，並列計算環境と統計計算 アルゴリズム，医学統計研究会秋季セミナー鹿児島 2011， 2011 年 9 月 3 日，鹿児島市

④黒石健太郎，越智義道，階層データへの一般化線形モデルの適応，大分統計談話会第 43 回大会，2011 年 2 月 17 日，大分市

⑤黒石健太郎，越智義道，階層線形モデルの推定とその計算，大分統計談話会第 42 回大会， 2010 年 10 月 21 日，大分市

⑥越智義道，Ｒによる離散データの解析：パッケージと開発環境，医学統計研究会 特定主題セミナー， 2010 年 6 月 12 日，東京都

⑦越智義道，統計的モデルの基礎-線形モデルを中心に， シンポジウム 2009「医療で必要とされる統計的基礎知識」， 2009 年 10 月 24 日，東京都

⑧越智義道，計算機統計学再考，医学統計研究会 秋季セミナーかごしま 2009， 2009 年 9 月 12 日，鹿児島市

〔その他〕
ホームページ等
http://bunsyo1.ad.oita-u.ac.jp:8080/kobetu.asp?id=227

６．研究組織
(1)研究代表者
越智 義道（OCHI YOSHIMICHI）
大分大学・工学部・教授
研究者番号：60185618

(2)研究分担者
小畑 経史（OBATA TSUNESHI）
大分大学・工学部・助教
研究者番号：00244153