科学研究費助成事業　　研究成果報告書

平成 ２６ 年　 ６ 月　 ３ 日現在

機関番号：１２６０１
研究種目：若手研究(B)
研究期間：2012～2013
課題番号：２４７０００２３

研究課題名（和文）スケジューリングと計算リソース量を柔軟に制御できる投機計算を考慮した分散計算環境

研究課題名（英文）A distributed computing environment that can handle speculation by controlling scheduling and computing resources

研究代表者
横山　大作（Yokoyama, Daisaku）

東京大学・生産技術研究所・助教

研究者番号：８０３４５２７２

交付決定額（研究期間全体）：（直接経費）　　3,400,000円、（間接経費）　　1,020,000円

研究成果の概要（和文）：ゲーム木探索に代表される大規模探索問題を解くために重要な、複雑かつ動的なスケジューリングに対応した分散計算プログラミング処理系の実現を目指した。十分な領域依存知識を用いた実アプリケーションであるコンピュータ将棋プレイヤ「激指」を用いて、処理系に望まれる機能を整理した。さらに、分散環境において動的なスケジューリングを可能にする処理系を構築し、実アプリケーションを用いた実験を通してその有効性を検証することができた。

研究成果の概要（英文）：This research proposed a technology for distributed computing programming framework that can reflect many complicated and dynamic scheduling strategies. Our research is mainly focusing on large scale search problems, such as Game-tree search. First we investigated a computer shogi player "Gekisashi" and clarified the requirements of real distributed search applications. Gekisashi is a state-of-the-art program with many domain specific knowledge. We then created a distributed programming framework for elastic computing that enables to schedule dynamically. We confirmed the effectiveness of our approach through evaluation.

研究分野：総合領域

科研費の分科・細目：情報学，ソフトウエア

キーワード：並列処理・分散処理　投機的計算　探索問題

１．研究開始当初の背景

組合せ最適化や経路探索など、実世界で大規模な探索問題を解きたいという要求は多い。枝限定法や確率的探索など、複雑な実問題を解くために利用されるアルゴリズムにおいては、全ての解の可能性をしらみつぶしに調べるのではなく、様々な技法を用いて探索範囲を限定し、可能性がなくなった解を動的に枝刈りして、総当たりよりはるかに効率よく探索を行うことが通常である。探索問題を分散計算する際には解の候補を複数の計算機で分担して探索を行っていくが、前述のような枝刈りの可能性がある計算の場合は、候補解の一部は探索の必要がないかもしれないタスクであり、いわば「投機的に」実行を試していることになる。

探索のように複雑な構造を持つアルゴリズムを大規模分散計算したいという要求に対し、ワークフローが記述できる Hadoop や Pig、Dryad、細粒度並列タスクが記述できる Cilk、Chapel、X10 など、これまでに多数のプログラミング環境が提案され、現在も盛んに研究されている。これらの言語では、クラウドコンピューティングの普及により、動的な環境や動的な計算制御への考慮が今後の重要な課題として認識されている[1]。しかし、これらのプログラミング環境は、問題を大量の部分タスクに分割し、それらを全て計算しきることを前提として設計されており、投機計算を十分考慮していない。

研究提案者はこれまでに、探索問題に適した分散プログラミングフレームワーク、動的にリソースを増減させる探索手法などを提案し、大規模な探索問題を分散計算環境で効率よく解くための基礎技術を提案してきた。これまでの研究過程で、投機計算を含む問題に対し、現在実現されている汎用のプログラミング環境では以下の 2 点が十分考慮されておらず、問題となっていることがわかってきた。

(1) 部分計算を行うスケジューリングを、計算全体の実行状況に応じて適切に設定しなくてはならない。優先的に探索するべき有望な解候補は探索の進行に伴って変化するため、探索のスケジューリングを随時適切に指示できなければ、計算の効率を大きく損ねる。Hadoop などはタスクの依存関係を考慮するのみであり、スケジューリングを指示する余地がない。また、比較的並列実行の制御の自由度が高い Cilk などでも、部分的にタスクを優先するなどの投機実行に必要な自由度は持たない。

(2)利用する計算リソース量と得られる解の質との適切なバランスをとる必要がある。Hadoop や Google App Engine などクラウド環境を指向した処理系では、基本的には並列実行可能なタスクが増えれば計算リソースを増大させる、という戦略をとる。しかし、例えばパーティクルフィルタのように確率的に有望な解を探索するアルゴリズムにおいては、ある程度の規模以上にリソースを増やしても解の改善度合いが減少し効果が見合わない状況が発生する。得られる解の質、探索実行状況などの動的な要素により、必要な計算リソース量をアプリケーション側から柔軟に指定できなければならない。

以上の 2 点の問題に対し、有効なプログラミング手法を構築することが求められている。

組合せ最適化問題、確率的探索など、本研究が対象としている大規模探索問題が現在実世界に山積しているが、Amazon EC2 などの低廉な分散計算環境が利用可能になりつつあるにも関わらず、並列プログラミングの難しさから求解を断念しているのが現実である。よって、本研究のような実用的な分散計算環境が社会に与えるインパクトは大きく、またその構築を通して別種の問題分野の計算についても多くの研究的知見が得られると考えている。一応用であるゲーム木探索に限定しても、大規模分散環境での「効率の良い・単純でない」探索を行なった例は世界的にも少なく、大量の計算機を用いて「強いゲームプレイヤ」を実現できれば、学術的にも社会的にも大きな影響を与えると考えられる。

国内外における研究動向として、クラウド環境のような動的大規模分散環境を意識した汎用プログラミングフレームワークは盛んに研究されているが、投機計算を十分考慮したものはまだ存在しておらず、投機計算を多用する探索問題への応用が困難な状況である。

２．研究の目的

本研究では、投機的計算を含む大規模探索問題を対象とし、スケジューリングと使用リソース量を柔軟に制御できる分散計算フレームワークを構築し、動的な計算機環境で大規模問題を解くための技術を開発することを目的とする。対象となる探索問題は確率的探索など多数存在するが、特に投機計算を多用するものにチェス、将棋などに用いられるゲーム木探索がある。これは、アルファベータ法と優秀なヒューリスティックを多数用いることにより、木の大部分を枝刈りして効率の良い探索を実現しており、分散計算においても、投機探索に対し複雑な制御を加えることによって高速化を図る研究が多数行われてきている[2]。そのため、本研究においてはゲーム木探索を例題とし、方式設計と評価に役立てていくこととする。

参考文献

[1] Cooper, B. F. et.al.: Benchmarking Cloud Serving Systems with YCSB, ACM Symposium on Cloud Computing, ACM, Indianapolis, IN, USA (2010)

[2] Hyatt, R.: The Dynamic Tree-Splitting Parallel Search Algorithm, ICCA Journal Vol. 20, No. 1, pp. 3-19 (1997)

３．研究の方法

本研究では、投機的計算を含む大規模探索問題に対象を絞り、動的な計算機環境で大規模問題を解くための分散計算フレームワークを設計、構築した。研究提案者のものも含めた従来研究では、投機的計算実現に重要な (1) 部分計算のスケジューリング、(2) 実行状況に応じた利用計算リソース量、の 2 点において十分な制御能力を持っていなかった。研究提案者は探索問題の実アプリケーションである将棋プレイヤ「激指」の開発を行ってきた。激指はコンピュータ将棋プレイヤとして世界トップレベルの性能を持っており、高度に最適化された探索順序制御アルゴリズムを利用しているため、分散計算においては (1) のスケジューリングが探索効率を大きく左右する。さらに、ゲーム木は探索する局面に応じて複雑さが大きく変化し、必要となるリソース量が数百倍程度のオーダで変化することも珍しくないため、(2) の動的なリソース量制御が求められる問題である。そこで、本研究では、ゲーム木探索を実アプリケーションの例として、提案フレームワークの設計、評価を行うこととした。具体的には、激指を利用して(1)(2) への要求を検証し、実問題において必要かつ十分な機能は何かを明確化し、その上で探索問題を対象とした応用範囲の広い分散計算フレームワークとして整理した。また、これらの機能に対しその効率的な実装方式を検討し、検証した。

４．研究成果

本研究の成果として、ゲーム木探索をアプリケーションとしたときの並列計算スケジューリングを詳細に検討し、実アプリケーションとして並列探索に望まれる機能を整理した。この過程において、ゲーム木探索の並列化に適した新しい探索アルゴリズムを提案し、有効性を検証することができた。

また、ゲーム木探索に必要な複雑なスケジューリング、特に動的なリソース量制御を実現する分散計算フレームワークを構築し、ゲーム木探索をアプリケーションとしてその有効性評価を行い、有用性を検証することができた。ここでは特に、消費電力の観点からも本分散計算フレームワークが有用であることが示された。

以下にそれぞれの成果の詳細を述べる。

(1) 並列探索の実アプリケーションとしてのゲーム木探索スケジューリングの検討と、より実用性を高めた新しい分散ゲーム木探索手法の提案

ゲーム木探索の分散計算を考える際、近年目覚ましく発展しているのがモンテカルロ木探索アルゴリズムに基づく手法である。特に囲碁に関しては従来の木探索アルゴリズムでは人間よりはるかに弱いプレイヤしか構築できなかったが、乱数を用いたモンテカルロ木探索を利用したアルゴリズムが提案され、急速に強さが向上している。これは、モンテカルロ木探索が比較的容易に分散計算化可能なため、コンピュータリソースを大量に利用して高速化できることも大きく影響している。この新たなアルゴリズムの成功を受けて、将棋においてもモンテカルロ木探索の適用が試みられてきたが、従来の木探索手法より良好な性能は得られていない。従来のモンテカルロ木探索手法(図 1)では、乱数を用いた適当な手順により終局まで局面を進める「プレイアウト」を多数繰り返して評価を行うが、ゲームの性質上、囲碁と異なり将棋では有効なプレイアウトを生成することが困難であること、および多数のプレイアウトによる評価が通常の木探索の正確さに及ばないという事情に依ると考えられている。特に後者については、ある局面において正解といえる手順がごく少数に限られ、その正解手順を長期間たどり続けて初めて局面に優劣が付くようなゲームに関して、現在のモンテカルロ木探索手法では効率よく探索空間を絞ることができないという問題点が露呈しているといえる。

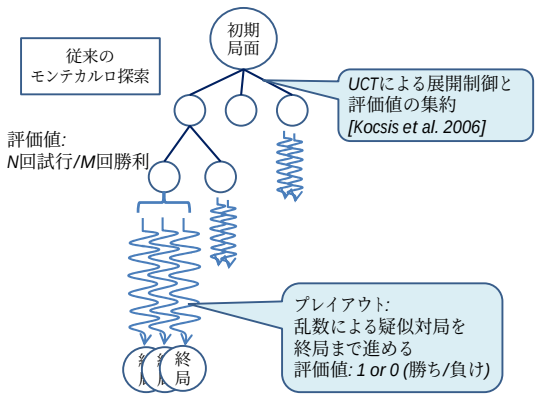

図 1: 従来のモンテカルロ木探索手法

これに対し、「乱数を付加した探索によるシミュレーション」、「Bayesian Approach による評価値伝搬」の 2 つの手法を利用したモンテカルロ木探索手法を提案した(図 2)。

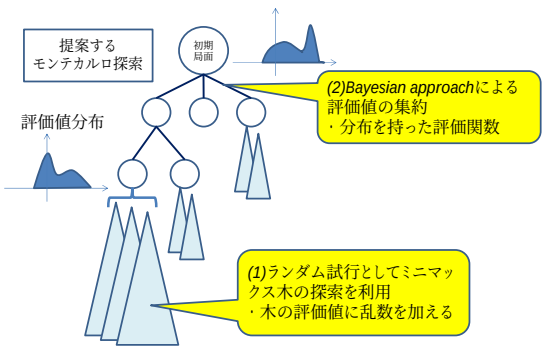

図 2: ベイジアンアプローチに基づいた木探索手法

本提案手法では、シミュレーション部分について、ランダムな指し手を生成するのではなく、評価関数に乱数を加えた探索を複数回

行い、その探索木で得られる指し手と評価値をシミュレーション結果とする。複数の探索により、木探索のリーフでは複数個の評価値が得られる。これをそのリーフの「確率分布を持つ評価値」として扱うことにする。また、探索木中の評価値伝搬手法として、リーフの値に確率分布があるときに、その確率分布を考慮したミニマックス探索を実現する手法である Bayesian Approach を利用する。

提案のモンテカルロ探索手法を実装し、オリジナルのプレイヤとの対局を多数回行うことで手法の性質と有効性を評価した。実装においては、コンピュータ将棋プレイヤ「激指」を利用し、その評価関数に模擬正規分布乱数を加えてシミュレーションを作成した。激指は実現確率打ち切り探索など既存の探索技法を多数導入した実用的なプログラムであり、世界コンピュータ将棋選手権などで優秀な成績を収めている。

テストでの勝率測定は、実戦棋譜の 31 手目の局面からランダムに 500 局面を選択し、その局面から先後を入れ替えて 1 回ずつ、合計 1000 対局によって求めた。

代表的なパラメタ設定を用いて提案手法の勝率と所要時間の関係を調べた。モンテカルロ木の深さパラメタ PVth を変化させたときの勝率を図 3 に示す。各系列はシミュレーション回数 Simnum を 1 から 7 まで変えた場合である。Simnum が 1、3 の系列では PVth を 0 から 12 まで、それ以外の系列は PVth を 0 から 8 まで変化させている。横軸は対局で消費した時間をオリジナルプレイヤとの比率で示している。いずれの設定においても、PVth を延ばしていくことで勝率は向上し、実験の範囲ではまだ限界には達していないように見受けられる。

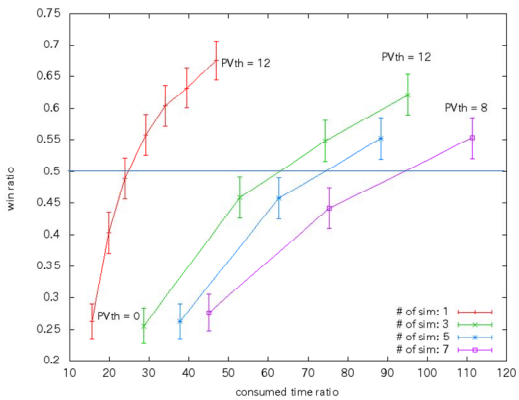

図 3: オリジナルプレイヤとの対戦実験における提案手法の勝率と所要時間の関係

図 4 はこの実験結果について、オリジナルプレイヤと比較した消費時間の比を横軸にしてプロットしたものである。シミュレーション探索の深さ Simdepth が 10 の場合は、8 の場合と比較して同一の PVth 設定での消費時間が 2.5〜3 倍程度延びており、同一の消費時間を要する領域で比較すると、Simdepth が 8 の方が高い勝率を得られる場合もあることがわかる。使用リソースに対する効率の良さと言う意味では必ずしもシミュレーションサイズを大きくした方が良いとは言い切れないと考えられる。また、シミュレーションサイズを大きくするということは、逐次実行部分の粒度を大きくするため、並列計算の適用を考える場合にはより不利になることが考えられる。

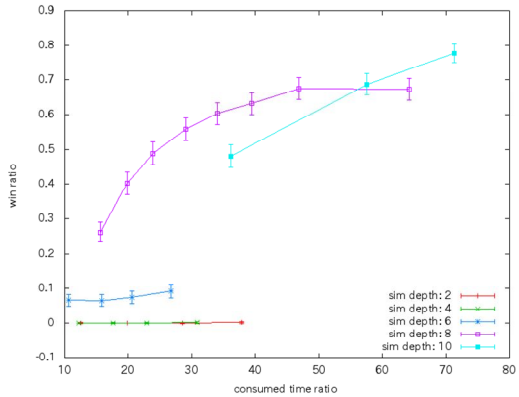

図 4: モンテカルロ木の深さを深くしたときの勝率と所要時間の関係

以上のように、オリジナルプレイヤとの対局実験を通して、並列実行に適したゲーム木探索手法である本提案手法によってコンピュータプレイヤの強さを向上させることができることを確認した。

(2) 動的なリソース量制御を実現するプログラミング環境の構築ならびにゲーム木探索アプリケーションを用いた評価

多数のユーザが利用するウェブサービス、センシングなどのアプリケーションにおいては、昼と夜など時間によってユーザ数が変化したり、大きなスポーツイベント、事故、天候変化などのイベントに伴って突発的に需要が発生するなどの要因で、サービス需要が大きく変動することがしばしば発生する。固定的なサービスインフラを利用していると、需要過多の時にはサービス品質が落ち、需要過小の時には無駄なリソース維持コストがかかるということで、このような需要変化を効率よく処理することができない。この問題に対し、オンデマンドに使用リソースを変動させてサービス提供を行う Elastic 計算と呼ばれる計算の重要性が高まっており、その効率の良い実行環境としてクラウドコンピューティングと呼ばれる計算環境が着目されている。しかし、プログラミング処理系、スケジューラなどのシステムがこれらの機能を積極的に利用する段階には至っていない。

我々は、このような Elastic な計算を可能にする分散計算プログラム処理系を構築し、探索を行う実アプリケーションを題材に、その有効性を確認した。また、特に重要な問題

である消費電力削減に対する効果を測るため、電源管理システムとプログラム処理系を連携させる実験を行い、Elastic な計算によって実計算環境においても消費電力量が 14% から 23%程度削減できることを確認した。

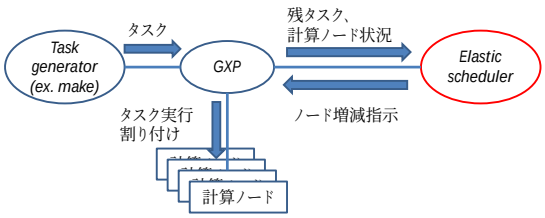

図 5: Elastic スケジューラの構成と GXP に必要な拡張

実アプリケーションの例として、コンピュータ将棋プレイヤ「激指」を用いた。将棋の探索は局面によって訪問すべき探索木の大きさが異なり、必要となる計算量が大きく変化する。我々は、計算量予測に基づき動的に計算に用いるリソース量を変化させることで、サービスレベルを守りつつ、必要最低限のリソースを利用して計算を行う手法を提案した。

激指は、動的な構成変化に対応するために、分散タスク処理系 GXP を利用する。GXP に対し、Elastic な計算を行うスケジューラの構築に必要な機能を追加し、Elastic 計算の制御システムとプログラム処理系との連携動作が行えるように修正を加えた。GXP には、ソケットやパイプなどのストリームを通してタスク投入を行うインタフェースが存在する。ここでは、そのインタフェースに

・計算機構成、タスク処理状況などの情報取得
・動的な計算機の参加・脱退

を可能とするような修正を加え、GXP の外部にスケジューラプロセスを別途作成し、接続した(図 5)。

このソフトウェアを用い、対局中継中の形勢解析サービスを想定したアプリケーションによる評価を行った。実験は、2010 年に行われた強豪将棋ソフトウェア対強豪女流プロの対局イベント、「清水市代女流王将 vs. あから 2010」での対局結果を用い、対局中に現れたそれぞれの局面(全 86 局面) で一定の深さまで探索することとした。サーバは 32 台までを利用することとし、制御方式として、

・const: 常に 32 台を利用する場合
・static: 各局面の探索に必要な時間をあらかじめ測定しておき、最も時間がかかる局面で 45 台を利用希望するように線形に必要台数を予測し、初期並列度ヒントを与えた場合。(なお、実際に計算に利用できるのは最大でも 32 台である。)
・dynamic: 探索総時間予測を行い、それをもとに 32 台までの範囲でヒントを与えた場合。

の 3 通りのスケジューラを構築し、計測を行った。static, dynamic については実験開始時には 3 ノードの計算ノードのみ電源が入っており、const は実験開始時から終了時まで常に 32 ノードが稼働状態である。

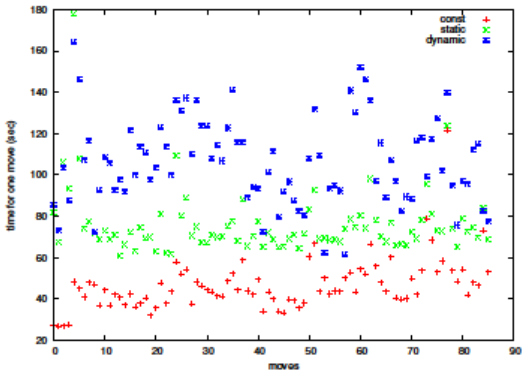

図 6: 手毎の探索所要時間

図 6 は、それぞれの方式での各局面での探索時間を示している。static はほぼ全ての局面で 60 秒から 120 秒の範囲で探索が終了しているが、const は探索時間が短い局面と長い局面の差が大きい。dynamic は必要なリソース量の予測が不正確な場合の制御例であり、static よりは探索時間が延びている。

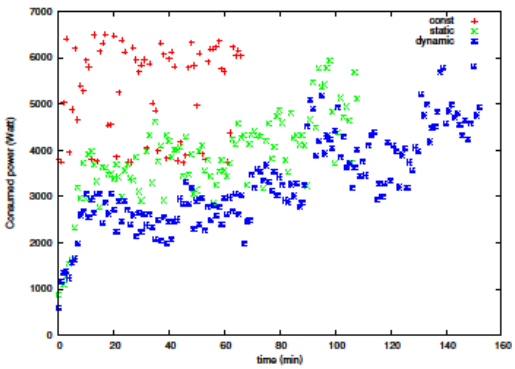

図 7: 消費電力の時間推移

また、それぞれの手法について、消費電力の時間変化を測定した結果を図 7 に示す。横軸は実験開始時からの経過時間(分) であり、1 分ごとに消費電力を測定した結果を縦軸にプロットしてある。図 7 では、計算ノードとネットワークスイッチの双方の消費電力の和を示している。const は常に 32 ノードの計算ノードを利用しているため、使用電力量の変化は CPU の DVFS などによる制御に起因している。比較的多くの時間において、4kW 程度の消費電力で稼働しているが、この時は計算需要と比較して投入リソース量が多すぎ、アイドルな計算機が多数発生していると考えられる。static、dynamic は総探索時間が const より延びているが、探索途中の消費電力は低く抑えられている。

図 7 の消費電力時間推移をもとに、計算全体で使用した積算電力量を求めたものを図 8 に示す。busy の系列が、各々の計算時間内で

の消費電力を示している。ここで、static な場合、すなわち解析すべき指し手がほぼ一定の時間間隔で到着した場合を想定すると、const では次の指し手の到着までは何もせず待つことになる。この時間に消費した電力量を計算し、idle の系列に示す。また、dynamic の場合は、指し手の到着に解析が追いついていない状況にあると想定される。

図 8 のように、計算自体に必要な電力量は const が最も少なく、電力制御を加えた方が大きい。しかし、アイドル時間の消費電力も含めて比較すると、Elastic な制御を行った方が全体の電力量が削減されていることがわかる。本想定では、制御を行わない場合に対し、完全な需要予測が行える(static) 場合はおよそ 14%、不完全な需要予測(dynamic) の場合には 23%程度の電力量削減が実現できた。

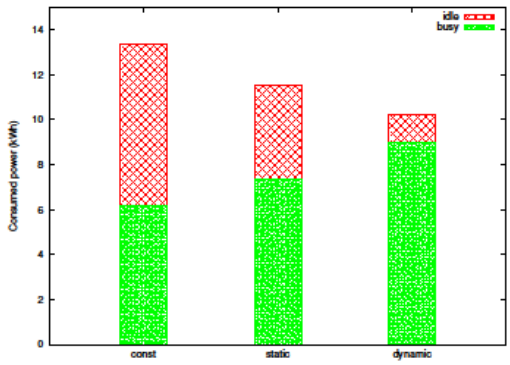

図 8: 積算消費電力

このように、動的なリソース量制御を実現できるようなプログラミング処理系を構築し、ゲーム木探索に関わる実アプリケーションを用いてその有効性を示すことができた。

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）

〔雑誌論文〕（計　1　件）

1. 横山 大作，喜連川 優，電力を考慮したアプリケーション構築のための計算機システムの提案，情報処理学会論文誌: コンピューティングシステム(ACS)，査読有，6(4)，pp. 72-82，2013. http://id.nii.ac.jp/1001/00095740/

〔学会発表〕（計　4　件）

1. 横山 大作，ベイジアンアプローチに基づくモンテカルロ木探索アルゴリズムの将棋への適用，情報処理学会　第 18 回ゲームプログラミングワークショップ，2013/11/8-10，箱根
2. 横山 大作，喜連川 優，電力を考慮したアプリケーション構築のための計算機システムの提案，第 11 回先進的計算基盤システムシンポジウム SACSIS2013，2013/5/22-24，仙台
3. 横山 大作，モンテカルロ木探索アルゴリズムの将棋への適用，情報処理学会第 17 回ゲームプログラミングワークショップ，2012/11/9-11，箱根
4. 横山 大作，田浦 健次朗，喜連川 優，電力を考慮したプログラミングのためのシステム構築に関する検討，情報処理学会　第 90 回プログラミング研究会，2012/8/1-3，鳥取

６．研究組織

(1)研究代表者

横山　大作（Daisaku Yokoyama）
東京大学・生産技術研究所・助教
研究者番号：80345272