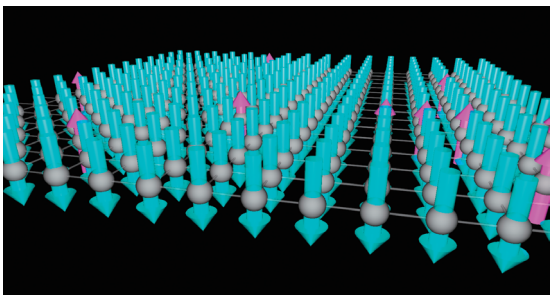
e-Science Journal
東京工業大学 学術国際情報センター
Global Scientific Information and Computing Center
Vol. 7

TSUBAME ESJ.



大規模半正定値計画問題に対する内点法アルゴリズムの高速計算
High-Performance General Solver for Extremely Large-Scale Semidefinite Programming Problems

古典スピン系における大規模GPUコンピューティング
The multiple GPU calculation for the classical spin model

大規模ウェブテキスト集合からの知識獲得とその応用
Knowledge Acquisition from a Large Web Corpus and its Applications

http://www.gsic.titech.ac.jp/TSUBAME_ESJ

大規模半正定値計画問題に対する内点法アルゴリズムの高速計算

藤澤 克樹*　遠藤 敏夫**
*中央大学＆JST CREST　**東京工業大学＆JST CREST

半正定値計画問題（SDP）は組合せ最適化、システムと制御、データ科学、金融工学、量子化学など非常に幅広い応用を持ち、現在最適化の研究分野で最も注目されている最適化問題の一つとなっている。また今後のエネルギー供給計画（スマートグリッド等）では非線形の複雑な最適化問題を扱う必要があり、これらの問題に対して強力な緩和値を算出できるSDPの高速計算技術の確立が急務とされている。SDPに対しては高速かつ安定した反復解法である内点法アルゴリズムが存在しているが、巨大な線形方程式系の計算が大きなボトルネックとなっている。著者らのグループでは内点法アルゴリズムを記述したソフトウェアの開発・評価・公開を15年以上行っており、疎性の追求、計算量やデータ移動量などによる計算方法の自動選択などの技術を他に先駆けて実現し、大規模な並列計算等によって上記のボトルネックの高速化と世界最大規模のSDPを高速に解くことに成功している。
今回、東京工業大学のスーパーコンピュータTSUBAME 2.0において、多数GPUの活用や計算と通信のオーバーラップ技術を応用することによって、主要なボトルネックの1つである線形方程式系のCholesky分解の高速化に成功した。さらに制約式の数が148万以上となる世界最大規模の巨大SDPを解き、SDPの世界記録の更新及び最大で533TFlops（Cholesky分解：4080 GPU）の性能を達成した。

1 最適化問題とソフトウェアに関する最新の傾向について

近年、最適化問題に対する研究の対象は数学的な理論からスーパーコンピュータによる大規模計算まで非常に多岐に及んでいる。また大規模かつ複雑な最適化問題を高速に解く需要は様々な産業界や学術分野において急速に高まりつつある。最適化問題に対するアルゴリズム及びコンピュータに関する研究は第二次世界大戦後から急速に発展し、今日までの約60年間は最適化問題に対するアルゴリズムとソフトウェアが連動しながら目覚しい発展を遂げてきた。しかし、初期のころは実社会から要求されるレベルに対して、アルゴリズムや計算機の能力が低かったため、実用に耐えうる複雑かつ大規模な最適化問題を扱うことができるようになったのは最近（21世紀以降）のことである。

実社会で要求される大規模最適化問題を解決するためには，短時間に膨大な計算量とデータ量を処理するための新技術が必要となる。例えば近年の日本は地震、津波、台風、洪水など大規模災害に何回も襲われており、大規模災害時には防災計画の策定、災害時の避難と誘導及び情報収集と解析、復興計画の策定、スマートグリッドによる高度かつ安定な電力供給などを行う必要があると言われている。このように大規模災害等で突発的に発生してリアルタイムに状況が変化し、かつ非常に計算量やデータ量などの規模が大きく従来の手法では処理が困難な性質を持つ実問題においては、大規模ネットワークの探索とクラスタリングの高速処理技術の開発が必須となる。つまり最近の国内外の情勢から大規模問題の解決が必要とされる分野には多くの新分野が加わり、その範囲も一企業や一国におけるレベルから、地球的規模に拡大しつつある。これらの諸問題の解決のためには新しい最適化計算の研究開発が必要という認識が共有されており、日本においても問題毎に理論からアルゴリズム、データ収集、ソフトウェア実装それに大規模計算までの研究者が集結して、多くの分野の融合を目的とした最先端理論（Algorithm Theory）+ 大規模実データ（Practice）+ 最新計算技術（Computation）による超大規模最適化問題の解決を目指す必要性が高まっている（図1）。

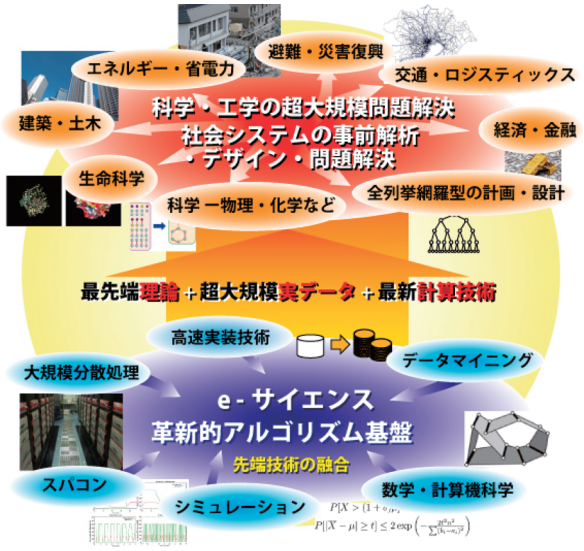


図1 最先端理論（Algorithm Theory）＋大規模実データ（Practice）＋最新計算技術（Computation）による超大規模最適化問題の解決

2 半正定値計画問題の現状

本解説では最適化問題として頻繁に用いられる数理計画問題の中から、近年研究の進展が特に目覚しく21世紀の線形計画問題として、その幅広い応用を期待されている半正定値計画問題（Semidefinite Program；SDP）を取り上げる。SDPなどの最適化

問題が最近特に注目を集めている理由には以下のようなものが考えられる。

1. 主双対内点法などのアルゴリズムによって多項式時間で最適解を求めることができる（つまり高速で安定したアルゴリズムが存在する）。

2. SDP は線形計画問題（LP）、凸二次計画問題や二次錐計画問題（SOCP）などを含んだより大きな凸計画問題の枠組であるが、SDPとして定式化できる最適化問題が解けるだけでなく、非凸最適化問題に対する強力な緩和値を導き出すことができる。そのためSDP を繰り返して解くことによって（最適に解くことが極めて難しいが実用上重要な）非凸最適化問題（例えば双線形行列方程式（BMI）など）を扱える可能性を持っている（つまり問題を適用できる範囲が広い）。

3. 組合せ最適化問題、整数計画問題、ノルムなどを用いた配置問題、システムと制御、ロバスト最適化、量子化学など非常に多くのSDPの応用が存在する（つまり非常に多彩な応用分野を持っている）（図2）。

4. 多くのSDPに対するソフトウェアが開発され、インターネットより公開されている[5,6,7,8,9,10,11,12]。さらに複雑で大規模な問題を解くためには、理論的成果を随時組み入れると共に、最新の並列計算技術（クラスタ&グリッド&クラウド計算）等との融合も必要不可欠である。最近では多くの成果が報告されていて相当大きな規模のSDPを解くことが可能になってきている（つまり公開されているソフトウェアで実際に大きな問題を解くことができる）。

SDPについての日本語の解説は[2,3,4]などを参照していただきたい。著者らのグループでは1995年からSDPに対する主双対内点法を実装したソフトウェアの開発と公開を行っている。現在ではSDPAのホームページ[1]から以下のソフトウェアの公開と自前の計算資源を持たなくてもWebからアクセスするだけでソフトウェアの実行が行えるSDPA Online Solver[2]の運用を行っている。

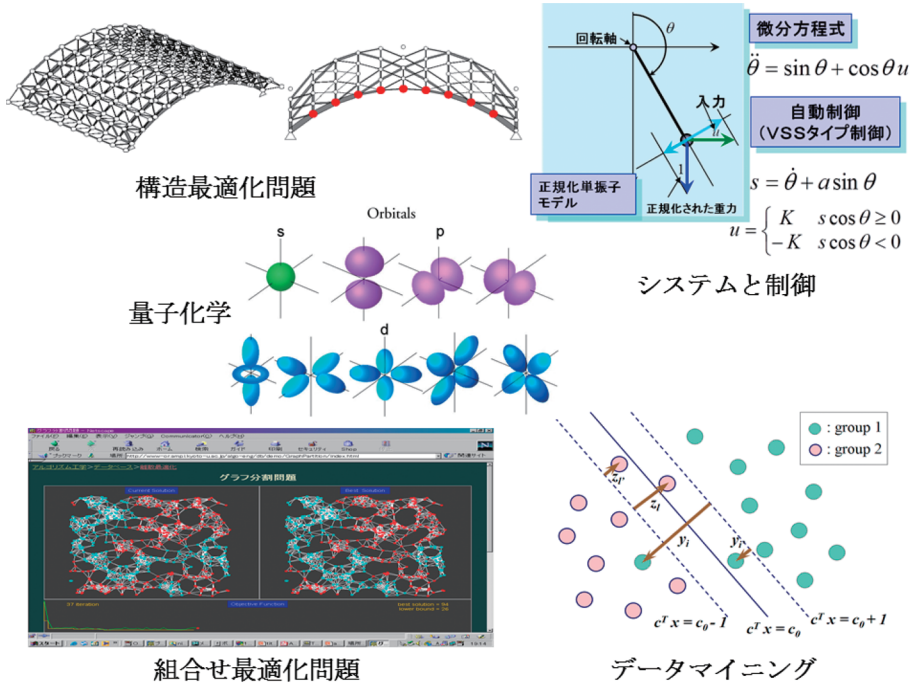


図2 SDPに関する主要な応用分野

[1] http://sdpa.indsys.chuo-u.ac.jp/sdpa/ [2] http://laqua.indsys.chuo-u.ac.jp/portal/

大規模半正定値計画問題に対する内点法アルゴリズムの高速計算

スーパーコンピュータ上での最適化問題の実行 3

GPU コンピューティングやクラウド・コンピューティングの普及によって、従来スーパーコンピュータで行われてきた数値実験等の一部が他の計算機環境でも実行されるようになっている。それでは最適化分野ではどのような最適化問題がスーパーコンピュータでの実行に適しているのであろうか。以下に適した条件を列挙してみよう。

1. 解くべき問題の規模が非常に大きいこと
2. アルゴリズムの演算量がある程度大きいこと
3. $\frac{\text{演算量}}{\text{データ移動量}}$ の比が十分程度大きいこと
4. 並列計算数に合わせて演算量やデータ移動量がほぼ均等になるように各プロセスに割り振ることができること
5. 並列に動作しているプロセス同士が高速かつ高い信頼性で頻繁に通信を行ったり、同期を取る必要があること

1と2については言うまでもないが、3は並列計算に限らず性能効率を上げるための必要条件である。4について説明すると、スーパーコンピュータではGPUでの並列計算と異なり、演算量やデータ移動量がほぼ均等になるように分割することができれば、原則的には各プロセスは全く異なる処理を行っても良いということである。また5の条件が必要でない場合、例えば単なるパラメータサーチで最後に結果だけ集めれば良いのであれば、クラウド・コンピューティングの技術等で大量の計算機資源を集めても良いだろう。反対に並列分枝限定法などでは5の観点からはスーパーコンピュータ上の方が実装が容易で、安定して動作させることができる。最近では各ノードにGPUを搭載したスーパーコンピュータが登場しているので、組み合わせが複雑になる分だけ実装は困難となるが、適切な演算量とデータ移動量の分割によって、さらなる性能向上も期待できる。

以上を踏まえてスーパーコンピュータでの実行に適した最適化問題について考えてみよう。TSP（巡回セールスマン問題）に対する適用例[3]などは知られているが、例えば線形計画問題（LP）に対する内点法では、入力データに疎性があることが多く、またベクトル演算が中心ということもあって、データを分割して転送するコストの比重が大きくなり大規模な並列計算の適用は難しい。また最短路問題に対するダイクストラ法ではアルゴリズム内の作業間の依存性が高く、並列して実行できる部分が少ないのでこれもまた適用が難しい。著者らのグループではSDPに対する主双対内点法アルゴリズムを並列化したソフトウェアSDPARA[5]の開発、公開を行っている。超大規模な SDP を SDPARA を用いて解いた場合には、上記の５条件を満たすことがわかっている。現時点では超大規模な SDP を解くためにはスーパーコンピュータが必須となっている。

すでに述べたようにSDPは組合せ最適化、システムと制御、データ科学、金融工学など幅広い応用を持っているが[2]、今回は超大規模SDPの需要を多く持つ量子化学分野に対する数値実験を紹介しよう（ソフトウェアはSDPARAを用いる）。化学または物理分野では原理的にはシュレーディンガー方程式を解くことによって、ほぼ全ての現象を理解することができると言われている。例えば、水分子の挙動、蛋白質の性質、光合成、超電導の仕組みなどがある。しかし、シュレーディンガー方程式を解くのは様々な面で困難を抱えるため2次の縮約密度行列の直接変分法という手法の開発、研究を行われている。著者らのグループではSDPARA[5,9,10,14]やSDPA-GMP[1]を用いて、世界で初めて正確に2次の縮約密度行列の直接変分を行い、多くの原子・分子に適用することに成功した[3]。特に以下のCH_3、NH_3、O_2から生じる巨大なSDPに対しては2010年3月に世界で初めて京都大学T2Kスーパーコンピュータを用いてSDPARAによって最適解を求めることに成功し、世界最大規模（2010年3月当時）のSDPを非常に正確に安定して解くことができた（表1）。今回の数値実験で解いた最も大きなSDPは図3のようなブロック対角構造を持っており（CH_3、NH_3から生じるSDP）、各行列の大きさは19,460 × 19,460、制約式の数は36,795 個、非零要素の総数は6,731,930個にも達する。

問題名 (分子名)	総計算時間 (秒)
H_2O	27,523.8
CH_3	68,593.4
NH_3	72,025.6
O_2	5,943.1

表1 量子化学分野の超大規模SDPに対するSDPARAの実験結果

[3] http://www.tsp.gatech.edu/

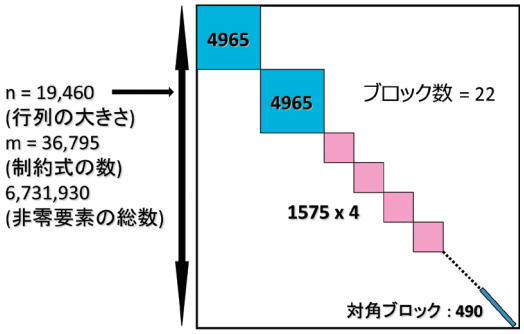


図3 2010年3月京都大学T2Kスーパーコンピュータで解いた超大規模のSDPとそのブロック対角構造

次に東京工業大学TSUBAME 2.0 スーパーコンピュータ[13]でのSDPARA の実行例を紹介する[14]。TSUBAME 2.0 は各ノードに2CPUと3GPUを搭載しており、主要なボトルネックの一つである線形方程式系のCholesky分解に対して、多数GPUの活用や計算と通信のオーバーラップ技術を応用することによって、制約式の数が148万以上となる世界最大規模の巨大SDP（図4）を解きSDPの世界記録の更新及び最大で533TFlops（Cholesky 分解：4080 GPU）の演算性能を達成した（2012年4月）。

1. 用いたSDP：QAP（二次割当問題）に対するDNN（Doubly Non-Negative）SDP緩和問題。制約条件数は約148万（図4）

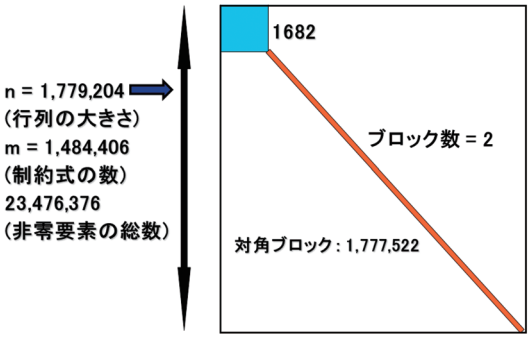


図4 2012年4月東京工業大学スーパーコンピュータTSUBAME2.0で解いた世界最大規模のSDPとそのブロック対角構造

2. 最大利用時の計算資源と実験結果

最大並列数：1360ノード、4080 GPU。なお、各ノードの3GPUにそれぞれ1MPIプロセスずつを割り当てたため、このときのMPIプロセス数は4080である。さらに各プロセスはGPUに加え、初期化・行列生成のためにOpenMPを介して複数CPUコアを用いている。この時1反復に要した時間は約2700秒、その中でCholesky分解部分は約2045秒である。結果的にその部分の演算性能は533TFlops（倍精度）となり、TSUBAME2.0を有効活用することにより、実用的な数理最適化問題においてペタスケールの演算を実現することができた。

東京工業大学のTSUBAME2.0スーパーコンピュータはCPU＋GPUのハイブリッド型であるが、京都大学T2Kスーパーコンピュータのように現在のスーパーコンピュータの主流はいわゆるスカラー型で各CPUは複数のCPUコアを有している。一昔前のスーパーコンピュータではベクトル型が主流であったために、過去にベクトル型用に作成されたソフトウェアは現在のスカラー型で性能を上げるためには大幅に書き換える必要もあるだろう。またMPIライブラリによって各プロセスに分割した後に複数のCPUコアを用いて、プロセス内をPthreadsやOpenMPなどを用いてマルチスレッド化する2段階並列が現在のスーパーコンピュータのアーキテクチャでは有効である。しかしCPUの演算性能はコア数の増大と共に急激に上昇しているが、CPUとメモリ間のバンド幅などはあまり向上が見られず、ハードウェア本来の性能を引き出すのが難しくなっている。よって、計算量とデータ移動量の正確な推定を行ったり、データの特性（疎性、サイズ）と性能値の関係を見極めることによって、2015年頃に登場するポストペタスケールスパコン、さらに2018年から20年頃に登場するエクサスケールスパコン等の超大規模スーパーコンピュータでも性能を発揮できるソフトウェアの作成を目指していくことになろう。

謝　辞

本研究はJST CRESTの研究領域「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」における研究課題ポストペタスケールシステムにおける超大規模グラフ最適化基盤、及び東京工業大学学術国際情報センター平成24年度春期グランドチャレンジ大規模計算制度の助成を受けています。

参考文献

[1] 藤澤克樹：高速化・最適化のためのBLAS入門、数学セミナー、588号、50-55、2010年9月.

[2] 藤澤克樹、梅谷俊治、応用に役立つ50の最適化問題、朝倉書店、(2009).

[3] 中田和秀、藤澤克樹、福田光浩、山下真、中田真秀、小林和博、最適化ソフトウェアSDPA、応用数理、Vol. 18、No. 1、2-14、(2008).

[4] 小島政和、土谷隆、水野眞治、矢部博、“内点法”、朝倉書店(2001).

[5] K. Fujisawa, K. Nakata, M. Yamashita, M. Fukuda: SDPA Project: Solving large-scale semidenite programs. J. Oper. Res. Soc. Japan, 50, 278-298 (2007) .

[6] B. Borchers: CSDP, A C library for semidenite programming. Optim. Meth. and Softw., 11 & 12, 613-623 (1999).

[7] J.F. Sturm: SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. Optim. Meth. and Softw., 11 & 12, 625-653 (1999).

[8] K.-C. Toh, M.J. Todd, R.H. Tütüncü, SDPT3 - a MATLAB software package for semidenite programming, version 1.3. Optim. Meth. and Softw., 11 & 12, 545-581 (1999)

[9] M. Yamashita, K. Fujisawa, M. Kojima: SDPARA: SemiDenite Programming Algorithm paRAllel version. Parallel Computing, 29, 1053-1067 (2003).

[10] M. Yamashita, K. Fujisawa, M. Fukuda, K. Nakata and M. Nakata, Parallel solver for semitenite programming problem having sparse Schur complement matrix, To appear in ACM Transactions on Mathematical Software (2012).

[11] I. D. Ivanov and E. de Klerk, Parallel implementation of a semidenite programming solver based on CSDP in a distributed memory cluster, Optim. Methods Software 25(3), 405-420, (2010)

[12] S. J. Benson, Parallel computing on semidenite programs, Preprint ANL/MCS-P939-0302 (2002).

[13] S. Matsuoka, T. Endo, N. Maruyama, H. Sato and S. Takizawa, The Total Picture of TSUBAME2.0. TSUBAME e-Science Journal, GSIC, Tokyo Institute of Technology, Vol. 1, 2-4 (2010)

[14] K. Fujisawa, T. Endo, H. Sato, M. Yamashita, S. Matsuoka and M. Nakata. High-Performance General Solver for Extremely Large-scale Semidenite Programming Problems, The Proceedings of the 2012 ACM/IEEE conference on Supercomputing, SC '12, 2012.

古典スピン系における大規模GPUコンピューティング

小村 幸浩* 岡部 豊*
*首都大学東京・理工学研究科

コンピュータの進歩により、大型、高速計算が可能となってきたが、特に最近、GPUの科学計算への応用が計算科学におけるホットな話題である。相転移現象などを、ミクロなモデルに基づき、統計力学の手法を用いて調べることができるが、そのような多体系の問題の数値的な手法として、モンテカルロシミュレーションが広く用いられている。その中で、クラスターフリップモンテカルロ法は、非常に有効な手法であるが、並列計算が容易ではない。本稿では、古典スピン系を対象として、Swendsen-Wang マルチクラスターアルゴリズムの大規模かつ高速なGPU計算を紹介する。特に、複数GPUを用いたアルゴリズムを、TSUBAME2.0上に実装し、効率のよい計算を実現したことを示す。

1 はじめに

私たちの身の回りにある物質は、ミクロ（原子レベル）な視点で見ると、アボガドロ数（6×10^{23}）程度の原子・分子からなっている。例えば、水の場合に、気体、液体、固体という３つの状態（３態）をとることは日常的によく知られているが、それぞれの状態においてアボガドロ数程度の水分子が存在している。温度、あるいは圧力などのマクロな変数を変えることにより状態が変化する現象を相転移という。ミクロな世界から出発して圧力、体積、温度などのマクロな世界の法則を示そうとする学問体系が統計力学であり、相転移もミクロな世界から出発して議論することができる。統計力学で相転移を考える際には、磁性体の相転移が取り扱い易い。鉄のような磁性体には、温度を上げると磁石の性質が消える相転移現象が存在する。

このような磁性体の相転移を扱う最も簡単な古典スピン系のモデルが、イジングモデルである（図1）。格子点上にスピン（ミクロな磁石）S_iをおき、このスピンは上向きか下向きをとり、変数としてS_iは±1の値をとるとする。イジングモデルのハミルトニアン（エネルギー）は

$$H = -J\sum_{<i,j>} S_i S_j, \quad S_i = \pm 1 \qquad (1)$$

である。J（>0）は交換相互作用、<i,j>は最近接での和をとる。このモデルは、最近接のスピンが平行ならエネルギー -J をとり、反平行ならエネルギー Jの値をとる。重要なことは、エネルギーが２つのスピンの値に依存し、相互作用が存在することである。

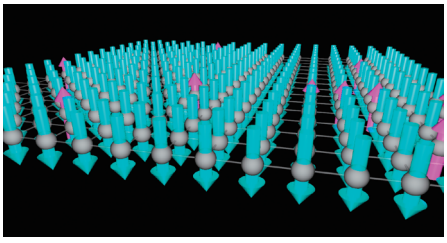


図1 2次元正方格子イジングモデル

統計力学で物理量の温度依存性を議論する際には、温度に依存した平均（ボルツマン平均）を求める。例えば、温度Tにおけるエネルギーは、

$$E(T) = <H>_T = \frac{\sum_{\{S_i\}} H \exp(-H/k_bT)}{\sum_{\{S_i\}} \exp(-H/k_bT)} \qquad (2)$$

の計算で求めることができる。ここでk_bはボルツマン定数であり、和の$\{S_i\}$は全ての状態に対する和をとる。取りうる全ての状態の和をとるということは、例えば、2500個の原子を扱う場合でも、$2^{2500} = 10^{752}$通りの計算が必要になり、直接計算するためには途方も無い時間が必要となる。なお、相互作用がなければこのような計算が簡単になる場合もあるが、イジングモデルのような簡単な相互作用でも、相互作用があると、特別な場合を除いて簡単化はできない。

そこで、統計力学の分野では数値解析手法として、確率を用いたマルコフ連鎖モンテカルロシミュレーションが発展してきた。マルコフ連鎖モンテカルロシミュレーションで、発生確率が（2）式のような温度Tのボルツマン分布となるように状態を作り出す手法をメトロポリス法[1]という。そのように状態を作り出せば、莫大な計算をすることなく、ある温度での物理量を単純平均で計算することができる。しかし、このメトロポリス法は、例えば、相転移点近傍のような興味ある温度領域では、平衡への緩和が非常に遅くなるなどの欠点もあり、それを克服する手法として、クラスターフリップ法[2,3]、マルチカノニカル法[4]などの手法が提案されてきた。コンピュータを用いてシミュレーションを実行するのは言うまでもないことであるが、コンピュータの発展がシミュレーションの役割を変えてきた。計算スピードの飛躍的発展で扱える系のサイズが大きくなり、また、より現実的な、より複雑な系が、シミュレーションの対象となってきた。それと同時に、アルゴリズムから解析方法を含む、計算手法の発展が常になされてきた。 近年は、GPU（graphic processing unit）の科学計算への応用が、計算科学のホットな話題となっており、TSUBAME2のシステムを効果的に使うことができれば、シミュレーションの分野における飛躍的な発展が期待できる。スピン系のモンテカルロシミュレーションのGPUを用いた計算に関しては、Preis et al.[5]がGPUによるメトロポリス法の計算を報告し、CPUを用いた計算に比べて大きな高速化を実現した。相互作用が短距離系のメトロ

ポリス法は、すべてが局所的な計算であり並列計算が容易であるのに対して、クラスターフリップ法は非局所的な計算で、GPUによる並列化は単純ではない。我々は、高速化計算、大規模計算が望まれていたGPUを用いたクラスターフリップ法の計算を実現したので、本稿で紹介する。1つのGPUを用いた計算、また、複数のGPUを用いた計算を解説し、TSUBAME2による実行結果に触れる。

クラスターフリップモンテカルロ法 2

初めに、クラスターフリップモンテカルロ法を説明する。これは、最近接のスピン同士を温度に依存した確率でボンドを繋げるか否かを決め、スピンのクラスターを作成し、各クラスター内のスピンを一度に更新させる大局的な数値手法である。この手法は従来問題だった臨界緩和の問題を劇的に和らげることに成功した[2]。本稿で取り扱うクラスターフリップ法はSwendsen-Wang（SW）マルチクラスターアルゴリズム[2]であり、このアルゴリズムはFortuin-Kasteleyn表現[6]を基に提案された。SWマルチクラスターアルゴリズムは大きく3つのステップに分かれており、それぞれのステップの計算の流れを図2に示す。図2では1メッシュ内に1スピンが存在するとし、スピン変数±1を0、1で表現している。それぞれのステップは以下の通りである。

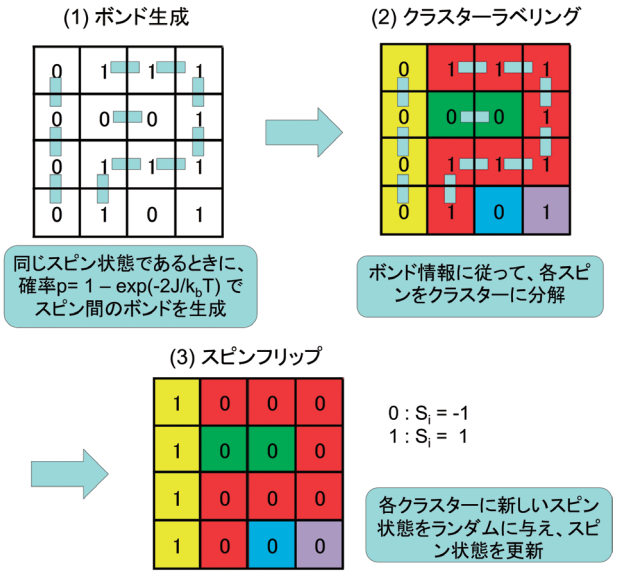


図 2 SW マルチクラスターアルゴリズムの手順

（1）隣接するスピンが同じ状態である場合に、温度に依存した確率を用いてスピン間にボンドを生成する。
（2）ボンド情報に従い、各スピンをクラスターに分類する。
（3）各クラスターの全てのスピンを新しいスピン状態にランダムに更新する。

このステップ（1）から（3）までの手順を1モンテカルロステップとし、繰り返しサンプリングを取ることで、各温度の物理量を測定する。ステップ（2）のクラスターラベリングをCPUで計算する際にはHoshen-Kopelman（HK）アルゴリズム[7]という Union-and-find アルゴリズム[8]の特殊な場合のアルゴリズムがよく用いられる。HKアルゴリズムでは各スピンに整数値のラベル番号を割り当てる。同じラベル番号のスピンは同じクラスターに属しており、クラスターを区別するため各クラスターには異なるラベル番号が与えられている。各スピンの適切なラベル番号を求める際には以下の様な手順を踏む。ラベル変数をlabel とする。各スピンのラベル番号とボンドで繋がっているスピンのラベル番号を比較し、一番小さいラベル番号に各スピンのlabelを更新する。その後、各スピンのラベルをlabel [label] と計算し、更新がなくなるまで計算を続ける。この際、小さなサイト番号のスピンから逐次的に計算を行う。この手順を踏むことで、各スピンは適切なクラスター（ラベル）番号を持つことができ、クラスターラベリングは完了する。

シングル GPU を用いた計算 3

クラスターフリップモンテカルロ法をCPU計算からGPU計算に拡張する場合、CPUでの逐次処理部分をどのように相互依存のない並列処理に変化させるかが重要となる。SWマルチクラスターアルゴリズムをGPU上で実現する際にも2章で示した3つのステップをとることに変わりはない。しかし、各々のステップにおけるCPU計算が直接GPU計算に適用出来るかを検討する必要がある。ステップ（1）のボンド生成はスピン状態を更新せずボンドのみを生成するため、各スピンが独立かつ同時にボンドを生成することができる。次にステップ（2）のクラスターラベリングは、CPU計算のHKアルゴリズムが逐次計算だからこそ成り立つ手法であるため、別の手法が必要になる。最後のステップ（3）のスピンフリップについては、事前に各クラスター（ラベル）番号毎に新しいスピン状態を用意することによって、各スピンを独立に更新することが出来る。以上から、ステップ（1）、（3）は容易にGPU化することができるが、ステップ（2）のクラスターラベリングは工夫をする必要がある。

シングルGPUでのクラスターラベリングは label equivalence algorithm[9]としてHawick et. al. に提唱されており、我々はクラスターラベリングアルゴリズムとして、そのlabel equivalence

algorithm[9]と Kalentev et. al. が提唱した改良アルゴリズム[10]の2つを用いて計算を行った[11]。このアルゴリズムの考え方はCPUのHKアルゴリズムと基本的には同じであり、各スピンがlabel[label]の計算を同時に実行する。HKアルゴリズムとは違い、この手法は一度のカーネルコールでは全てのクラスターラベリングが完了しないため、更新がなくなるまでカーネルコールを繰り返す。この手法の詳しい説明は参考文献[11]にあるので、ここでは割愛する。

次に、GPUとCPUでのSWマルチクラスターアルゴリズムの計算性能の比較を行う。モデルとして2次元正方格子イジングモデルを使用する。また、クラスターの大きさは温度に依存するため、転移温度上 $k_bT_c/J= 2.269...$[12]における計算時間を比較する。比較するCPUとして Intel(R) Xeon(R) CPU W3680 @ 3.33GHzの1コアを使用し、GPUとしてはNVIDIA Geforce GTX580を使用した。コンパイラとして、CPUではgcc 4.1.2にオプション –O3 を使用し、GPUはCUDA4.0を使用した。1列のスピンの個数をLとし、2次元正方格子L×LのL=256,1024,4096の結果を表1 に示す。表1では1モンテカルロステップ当たりの計算時間（ミリ秒）を記載している。CPUの手法としては2章で説明したHKアルゴリズムを使用した。表1より全てのサイズでCPUより大幅に高速化されていることがわかる。またHawick et. al.のアルゴリズムよりもKalentev et. al.のアルゴリズムの方が速く、Kalentev et. al.のアルゴリズムは改良されていることが確かめられた。

	L=256	L=1024	L=4096
GTX 580 Hawick et. al.	0.33 msec	3.16 msec	49.8 msec
GTX 580 Kalentev et. al	0.31 msec	2.71 msec	42.1 msec
Xeon(R) W3680	1.89 msec	32.82 msec	523.4 msec

表1 SWマルチクラスターアルゴリズムでのGPUとCPUの1モンテカルロステップ当たりの計算時間（ミリ秒）

マルチ GPU を用いた計算 4

シングルGPUで扱える系のサイズは限られている。TSUBAME2のようなマルチGPUのシステムを用いれば、非常に大きな系を直接的に扱うことができ、計算の高速化が実現できる。GPU計算自体が並列計算であるが、それを更に複数のGPUで計算を実行する際には、考慮すべき問題がある。まず、複数のGPU間の通信が必要である。また、シングルGPUにおける計算が共有メモリ型の並列計算であるのに対して、マルチGPUでの計算は分散メモリ型の並列計算である。この違いは今回のSWマルチクラスターアルゴリズムをマルチGPUに拡張する際に大きな問題となる。

具体的に、計算するモデルとして2次元正方格子イジングモデルを取り上げる。GPU間のデータ通信の際にはGPU-CPU間、CPU間、CPU-GPU間の3段階の通信が必要となり、CPU-GPU間の通信ではCUDAのAPIを使用し、CPU間の通信ではMPIライブラリを使用する。各GPUの配置を図3のようにとる。各GPU内には各領域のスピン状態と境界外のスピン状態の情報も持たせる。境界外のスピン状態を各GPUに持たせることでステップ（1）のボンド生成の計算は最近接GPUとの通信を行わず実行できる。ステップ（2）のクラスターラベリングは、まず、各GPU領域でラベリングが終了した後に全体のGPUのラベリングを行う2段階のラベリングの方法を採用する。この全体のGPUのラベリングの際には、分散メモリ型の対応が必要であるが、それについては、次に議論する。ステップ（3）でのスピンフリップにおいても分散メモリ型のための考慮が必要となる。

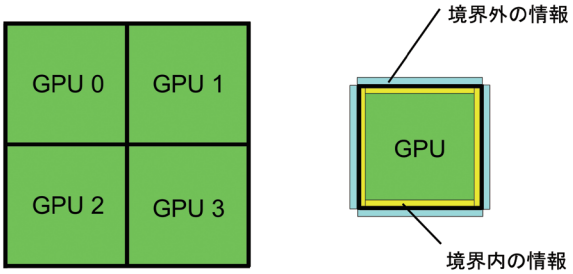


図3 各 GPUの配置と各 GPUの担当領域

シングルGPUの手法をマルチGPUに直接適用させるとすると、分散メモリ型への移行のため、ステップ（2）、（3）の方法で不都合が生じる。ステップ（2）のクラスターラベリングは各スレッドがlabel[label]を更新がなくなるまで実行するが、これは共有メモリ型だから出来た手法である。分散メモリ型の場合には、label[label]のメモリアクセスの際、異なるGPUのメモリにアクセスする可能性があるため、全体のGPUのラベリングの際にこの方法では計算が出来ない。また、ステップ（3）のスピンフリップでは、各GPUは適切な新しいスピン状態を取得し、各GPUの領域内のスピンを更新させる必要がある。つまり、シングルGPUの計算で用いた手法をそのまま適用すると、各GPUは自身の領域内にあるクラスター番号を調べ、クラスター番号から新しいスピン状態があるGPUを探し、GPU間で新しいスピン状態を通信するというプロセスが必要となる。しかし、これには莫大な時間が必要となる。

複数のCPUを使用した分散メモリ型のクラスターラベリングアルゴリズムとしては、マスタースレイブの方法[13]やグローバルラベル配列表を作成する方法[14]、boundary情報を最近接のCPUに通信し、更新がなくなるまで通信を続ける方法（relaxation method）[15]などが提案されている。またboundary情報を送る方法としては、ローカルなルートを持たせる方法[15]や、スピンの重複を作成し重複したスピンのラベル番号のみを更新させる方法[16]などがある。一つ一つのGPUのメモリ容量はそれほど大きくないため、我々は、このboundary情報を通信し、更新がなくなるまでの通信を続ける方法（relaxation method）を採用する。

また、ラベリングの際に用いるラベルとしてローカルラベルとグローバルラベルの2変数を用い、各GPU内でのラベリングの際にはローカルラベルを使用し、全体のラベリングの際には2つのラベルを併用させグローバルラベルを更新させる。その際、全GPUのグローバルラベルの更新がなくなるまで、最近接GPUとのboundary情報の通信を絶えず行う。また、その際にGPU間での通信時間の隠蔽も行なっている。ステップ（3）でのスピンフリップでは、新しいスピン状態をクラスター番号内に含ませ、ビット操作を用いてラベル情報から抜き出すことにより、分散メモリの問題を解決した[17]。

図4にTSUBAME2.0上でのマルチGPUを用いたSWマルチクラスターアルゴリズムの計算性能のグラフを示す。GPUはNVIDIA Tesla M2050であり、コンパイラとしてはCUDA4.0とopenMPI 1.4.2を使用している。また、それぞれの点は2次元正方格子イジングモデルの転移温度上での計算性能である。グラフは両対数でプロットしており、各GPUの格子サイズは1024×1024、2048×2048、4096×4096の場合をプロットしている。クラスターラベリングアルゴリズムはループアルゴリズムであるため、GFLOP数を正確に求めることが出来ないのと実用的な面を考え、縦軸は1ナノ秒当たりのスピンフリップ数（全体の格子サイズ/計算時間（ナノ秒））、横軸はGPU数をとっている。グラフは全体の格子サイズを一定にした強スケーラビリティと、グラフの挿入図では各GPUの格子サイズを4096×4096で固定した弱スケーラビリティをプロットしている。グラフの挿入図内の点はほぼ直線上に並んでおり、その係数を見積もった所 0.91 となりGPU数を増やした際の計算効率の向上がほぼ理想的であることがわかる。最後に実際の計算時間を示すと、16GPUを使用した16384×16384の計算時間は1モンテカルロステップ当たり0.09秒であり、256GPUを使用した65536×65536の計算時間は1モンテカルロステップ当たり0.13秒である。

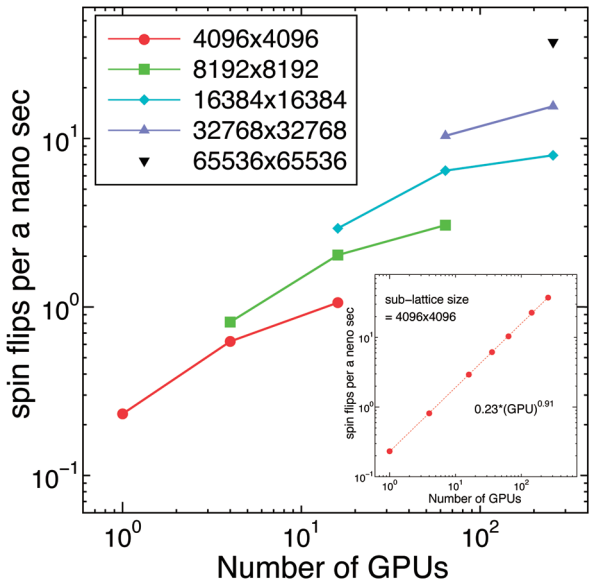


図4 TSUBAME2.0上でのGPU数と計算性能の関係

おわりに 5

我々は古典スピン系でよく用いられるモンテカルロアルゴリズムの一つであるSWマルチクラスターアルゴリズムについて、シングルGPUならびにマルチGPUの計算方法の提案を行った。CPU計算からシングルGPU計算へ拡張する際、CPUのクラスターラベリングアルゴリズムは直接適用出来ない。そこで、我々はHawick et. al. が提唱したlabel equivalence algorithm[9]と Kalentev et al. が提唱した改良アルゴリズム[10]を用いてシングルGPUにおけるSWクラスターアルゴリズムを実現した。結果、全ての格子サイズでGPU計算がCPU計算より高速化されることを示した。

次に我々は、シングルGPUからマルチGPUへのSWマルチクラスターアルゴリズムの拡張を行った。この拡張の際には、共有メモリ型から分散メモリ型へ変化するため、クラスターラベリングアルゴリズム部分とスピンフリップ部分において、シングルGPUにおける手法を直接適用することは出来ない。そこで我々は、relaxation methodと2つのラベル変数を用いる工夫を行い、2段階のクラスターラベリングの方法を採用することにより、マルチGPUのクラスターラベリングの問題を解決した。また、クラスター番号内に新しいスピン状態をビット操作を用いて含ませることにより、スピンフリップの分散メモリの問題も解決した。マルチGPUを用いたSWマルチクラスターアルゴリズムをTSUBAME2.0に実装し、計算性能を評価した結果、クラスターアルゴリズムという非局所的な問題であるにもかかわらず、スケーラビリティの

よい効率的な計算が実現できていることを確認した。

マルチGPUのSWクラスターアルゴリズムは、幅広いスピン系の問題に適用可能で、2次元古典XYモデルの大規模計算[18]などへの応用を行っている。さらに、開発したマルチGPUを用いたクラスターラベリングはこのSWマルチクラスターアルゴリズムのみならず、パーコレーションや画像処理といった問題にも適用できる。今後はこの手法の改良に加え、他分野への応用も視野にいれ開発を続けていく予定である。

謝　辞

有益な議論をしていただいた、富田裕介、青木尊之、Wolfhard Janke の各氏に感謝する。日本学術振興会科学研究費補助金の補助に感謝する。計算は、東京工業大学 TSUBAME2.0のシステムを用いた。

参考文献

[1] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth and A.H. Teller, E. Teller: Equation of State Calculations by Fast Computing Machines, J. Chem. Phys. Vol. 21, pp. 1087-1092 (1953)

[2] R.H. Swendsen and J.S. Wang: Nonuniversal critical dynamics in Monte Carlo simulations, Phys. Rev. Lett. Vol. 58, pp. 86-88 (1987)

[3] U. Wolff: Collective Monte Carlo Updating for Spin Systems, Phys. Rev. Lett. Vol. 62, pp.361-364 (1989)

[4] F. Wang and D.P. Landau: Phys. Rev. Lett. Vol. 86 pp. 2050-2053 (2001).

[5] T. Preis, P. Virnau, W. Paul and J.J. Schneider: GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model, J. Comp. Phys. 228 pp. 4468-4477(2009).

[6] C. M. Fortuin and P. W. Kasteleyn: On the random-cluster model: I. Introduction and relation to other models, Physica Vol 57, pp. 536-564 (1972)

[7] J. Hoshen, and R. Kopelman: Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm, Phys. Rev. B. Vol. 14, pp.3438-3445 (1976).

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein: Introduction to Algorithms, MIT Press, third edition, 2009.

[9] K.A. Hawick, A. Leist and D. P. Playne: Parallel Graph Component Labelling with GPUs and CUDA, Parallel Computing Vol. 36, pp. 655-678 (2010)

[10] O. Kalentev, A. Rai, S. Kemnitzb and R. Schneider: Connected component labeling on a 2D grid using CUDA, J. Parallel Distrib. Comput. Vol. 71, pp. 615-620 (2011)

[11] Y. Komura and Y. Okabe: GPU-based Swendsen-Wang multi-cluster algorithm for the simulation of two-dimensional classical spin systems, Comp. Phys. Comm. Vol. 183, pp. 1155-1161 (2012)

[12] L. Onsager: Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition, Phys. Rev. Vol. 65, pp. 117-149 (1944)

[13] M. Bauerfeind, R. Hackl, H.G. Matuttis, J. Singer, I. Morgenstern, 3D Ising model with Swendsen–Wang dynamics: a parallel approach, Physica A Vol. 212, pp. 277–298 (1994)

[14] J.M. Constantin, M.W. Berry, B.T. Vander Zanden, Parallelization of the Hoshen–Kopelman algorithm using a finite state machine,Internat. J. Supercomput. Appl. High Perform. Comput. Vol. 11, pp. 34–48 (1997)

[15] M. Flanigan, P. Tamayo, Parallel cluster labeling for large-scale Monte Carlo simulations, Physica A Vol. 215, pp. 461–480 (1995)

[16] J.M. Teuler and J.C. Gimel :A direct parallel implementation of the Hoshen–Kopelman algorithm for distributed memory rchitectures Comp. Phys. Comm. Vol. 130, pp. 118-129 (2000)

[17] Y. Komura and Y. Okabe: Multi-GPU-based Swendsen–Wang multi-cluster algorithm for the simulation of two-dimensional q-state Potts model, Comp. Phys. Comm. Vol. 184, pp. 40-44 (2012)

[18] Y. Komura and Y. Okabe: Large-Scale Monte Carlo Simulation of Two-Dimensional Classical XY Model Using Multiple GPUs, J. Phys. Soc. Jpn., Vol.81, 113001 (2012).-

大規模ウェブテキスト集合からの知識獲得とその応用

河原 大輔*　黒橋 禎夫*
*京都大学大学院情報学研究科

SF小説、映画では、ロボットが人間と自然にコミュニケーションをしている。このようなロボットを実際に実現するには、ロボットが言語を理解する必要があり、このためには人間がもっているような常識的な知識をロボットに与えることが不可欠である。しかし、そのような知識は膨大であるが故に、人手で記述するのは難しく、長い間、行き詰まりをみせていた。しかし、ウェブを中心とする近年の計算機ネットワークの進展に伴い、膨大な量のテキストが集積されるようになり、これを用いて知識を獲得することが可能になった。本稿では、大規模ウェブテキストからの言語知識の自動獲得と、得られた知識に基づく言語解析について紹介する。

1 はじめに

近年、音声インターフェースがスマートフォンなどに搭載され、特定のタスクでは実用的に使われるようになった。しかし、コンピュータと人間が自然で意味のあるコミュニケーションを行うにはいまだ程遠い状況である。このようなコンピュータを実現するためには、人間に近いレベルで言語を理解し、使いこなす計算機システムを実現する必要がある。

次の二文は、日本語文の構造的曖昧性に関する古典的例文である。

(1) a. クロールで泳ぐ少女を見た
　　b. 望遠鏡で泳ぐ少女を見た

この二文は同じ文体であるが、「クロールで/望遠鏡で」の修飾先(構文構造)が異なる(下線部が二重下線部を修飾する)。このような曖昧性は、日本語の文法だけでは解消することができず、語と語の関係に起因する現象を正確に扱う必要がある。そのためには、「クロールで泳ぐ」「望遠鏡で見る」のような語と語の関係に関する常識的な知識が必要となるが、これは膨大であるため、人手で記述しつくすことは極めて困難である。

近年、膨大な量のテキストデータがウェブから得られるようになってきており、こうしたテキストデータからの自動獲得により、常識的な知識を得る研究が進みつつある。我々は、ウェブから収集した超大規模テキスト集合から上記のような言語知識を自動獲得し、それに基づく解析システムを開発している。本稿では、TSUBAMEの大規模並列計算資源を利用して、この知識獲得プロセスを極めて短期間で実現したことを紹介する。

獲得する言語知識としてはさまざまなものがあるが、本稿では、「誰が何をどうした」のような述語項構造と呼ばれるものを集約した格フレームについて紹介する。これはもっとも基本的な常識的な言語知識の一つである。格フレーム獲得の具体的な手順は以下の通りである[1]。

1. ウェブページ集合からの日本語文の抽出(コーパスの作成)
2. コーパスに対する構文解析
3. 構文解析結果のフィルタリング、クラスタリングによる格フレーム獲得

今回は、他のクラスタ計算機上で作成したコーパスをTSUBAMEに転送し、このコーパスに対する構文解析処理と格フレーム獲得処理をTSUBAME上で行った。

以下では、まず格フレームの自動獲得手法について説明し、次に格フレームに基づく解析について述べる。その後、TSUBAME上における実験結果について報告する。

2 格フレームの自動獲得

格フレームとは、述語の各用法ごとに、述語とそれが関係をもつ項(名詞)を集約して記述したものである。たとえば「焼く」という述語の格フレームのひとつとして次のようなものが考えられる。

(2) {私、人、…} が　{オーブン、フライパン、…} で
　　{パン、肉、…} を　焼く

このような格フレームは、大量のコーパスを構文解析し、その解析結果から構文的曖昧性のない述語と項の関係を抽出、クラスタリングすることによって自動的に獲得することができる[8]。構文的曖昧性のない述語と項の関係とは、構文解析器における文法規則によって係り先述語の候補がただ1つしかない項を表す。たとえば、次に挙げた例では、○下線部が構文的曖昧性のない項として抽出される。

(3) a. 今日は石窯で○ パンを○ 焼いています。
　　b. 80種類ものパンを○ 焼いていますが、…
　　c. その後、パンを× 焼いた余熱を利用し…
　　d. 直径が× 15センチのケーキを焼きます

（3a）、（3b）では、文末および「〜が」が強い区切りと認識され、「焼く」に曖昧性なく係る「石窯でパンを」と「パンを」がそれぞれから抽出される。

一方、（3c）、（3d）の×下線部は、係り先の曖昧性があるため抽出されない。（3c）では、下線部「パンを」は、「焼いた」に係ると正しく解析されるが、「焼いた」と「利用し」の2つの係り先の曖昧性をもっているため抽出しない。（3d）では、下線部「直径が」が「15センチ」と「焼きます」の2つの係り先の曖昧性をもっているため同様に抽出しない。解析器は、「直径が」が「焼きます」に係ると誤って解釈してしまうため、このような関係を抽出すると誤った格フレームを作ることになる。この処理は、述語のように振る舞う句を文法規則によりきちんと把握しているために可能となっている。

格フレーム獲得における大きな問題は述語の多義性であり、「焼く」の場合では「パンを焼く」「手を焼く」のような意味、用法の異なる表現を区別してクラスタリングを行う必要がある。この問題に対して、述語の直前の項が述語の用法の決定に強く影響していると考え、述語とその直前の項を組にしたものを単位としてクラスタリングを行う。たとえば「焼く」の場合、「パンを焼く」「肉を焼く」「手を焼く」などの組で扱うことによって、意味、用法を区別することができ、格フレームがそれぞれに対して獲得される。さらに、語の類似度に基づくクラスタリングを行うことによって、「パンを焼く」「肉を焼く」のような類似している格フレームをまとめることにより、最終的に表1のような格フレームが得られる。

	格	用例（数字は頻度を表す）
焼く（1）	ガ ヲ デ	私：18，人：15，職人：10，… パン：2484，肉：1521，ケーキ：1283，… オーブン：1630，フライパン：1311，…
焼く（2）	ガ ヲ ニ	先生：3，政府：3，人：3，… 手：2950 攻撃：18，行動：15，息子：15，…
焼く（3）	ガ ヲ ニ	メーカー：1，ディストリビューター：1，… データ：178，ファイル：107，コピー：9，… R：1583，CD：664，CDR：3，…
⋮	⋮	⋮

表1 自動獲得した格フレームの例

3 格フレームに基づく構文・格解析

このような大規模格フレームを構文解析に統合することによって、最初に挙げた例（1a）と（1b）の構文構造の曖昧性が正しく解消できるようになる[2]。たとえば、（1a）を正しく解析するために必要な「クロールで泳ぐ」という関係は、京大コーパス[3]のような数万文規模の新聞記事コーパスからは獲得できないが、大規模コーパスから獲得した格フレームには含まれている。つまり、格フレームには、自立語間の振る舞いに関する知識が網羅的に獲得できており、これを利用することによって例（1）のような構文的曖昧性を精度よく解消できるようになった。

また、格フレームを利用することによって、構文構造だけではなく、係助詞句や連体修飾詞の述語に対する格の情報を同定することができる。たとえば、例（1）のどちらの文においても「少女」は「泳ぐ」に対して「少女が泳ぐ」という関係であるが、これは格フレームに基づく格解析によって同定している。このように、大規模なコーパスから自立語間の振る舞いを学習し、解析に役立てる手法は、英語など他・多言語に関しても適用されるようになってきている[4、6]。

4 実験結果

まず、約30億件のウェブページから日本語文を抽出、重複を除去し、日本語約150億文からなるコーパスを作成した。この作成は他の計算機クラスタで行ったが、その結果をTSUBAMEに転送した。TSUBAME上で、このコーパスに対して構文解析を適用した。この処理を行うために、約20万CPUコア・時の計算を要した。この処理の最小単位は「文」であり、1万文程度の細かい処理単位に分割することによって、embarrassingly parallelに実行することができた。

次に、2節で述べた格フレーム獲得手法を用いて、150億日本語文の構文解析結果から格フレームを獲得した[1]。その結果、約4万述語からなる格フレームが得られた。この処理はTSUBAMEで行い、約10万CPUコア・時の計算を要した。また、得られた格フレームを解析器に組み込むことによって、構文・格解析器を作成した。

さらに、コーパスのサイズと解析器の精度との関係を調べるために、上記150億文コーパスからサンプリングすることによって、150万文、600万文、2500万文、1億文、4億文、16億文、64億文のコーパスを作成した。それぞれのコーパスからの格フレーム獲得とそれに基づく解析器の作成を行い、コーパスのサイズが大きくなるにつれて、解析器の精度が向上することを確認した（図1）。図における「格フレームのカバレッジ」は、テスト文における述

語項構造がどれだけ格フレームに含有されるかであり、こちらもコーパスのサイズが大きくなるにつれて高くなることを確認した。なお、図における「同義句認識」は、「景気が悪化する⇔景気が冷え込む」のような句レベルの同義関係の認識[9]であり、「省略解析」は日本語文中において省略されている主語などの名詞を同定する解析[7]である。

上記計算のTSUBAME上での実行には、東京大学田浦研究室にて開発されたグリッドシェルGXP[2]を用いた。また、TSUBAMEのキューとしては、主に予約キュー（H キュー）を用い、最大で300 ノード（3,600CPUコア）を並列に利用した。

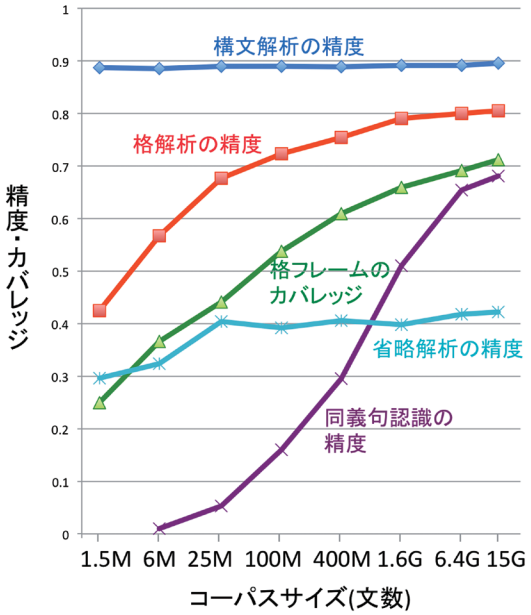


図1 コーパスサイズを増やしたときの精度・カバレッジの変化

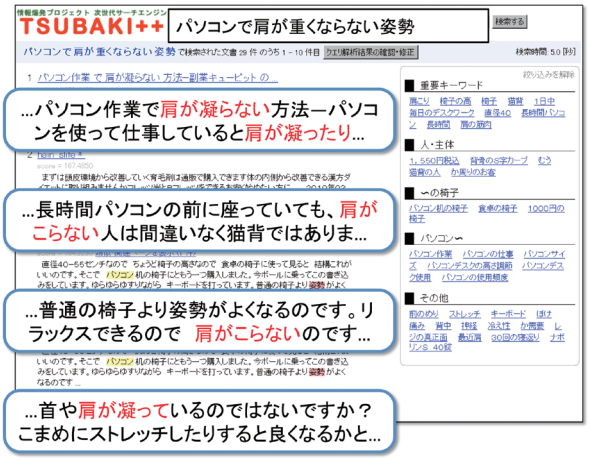


図2 検索エンジンTSUBAKIによる検索例

おわりに 5

本稿では、ウェブから収集した超大規模テキスト集合からの格フレーム自動獲得および格フレーム基づく構文・格解析について述べた。格フレーム獲得処理は、TSUBAMEの大規模並列計算資源を利用して、極めて短期間で達成することができた。

自動獲得した言語知識や構文・格解析システムに基づくアプリケーションとして、検索エンジンTSUBAKIを開発している[5]。TSUBAKIは、図2のように、特に自然文のクエリを入力した場合に、同義知識や言語構造を生かした検索ができることが特徴である。このように、大規模に獲得した言語知識や深い言語解析は、実際のアプリケーションにおいて有効であることが示されている。

今後は、精度・カバレッジについて、コーパス規模の拡大によりさらなる改善が予想されることから、さらに規模を拡大して知識獲得に取り組む予定である。その際には、さらに大規模にTSUBAMEを利用することを考えている。

参考文献

[1] Daisuke Kawahara and Sadao Kurohashi. Case frame compilation from the web using highperformance computing. In Proceedings of LREC2006, 2006.

[2] Daisuke Kawahara and Sadao Kurohashi. A fully-lexicalized probabilistic model for Japanese syntactic and case structure analysis. In Proceedings of HLT-NAACL2006, pp. 176-183, 2006.

[3] Sadao Kurohashi and Makoto Nagao. Building a Japanese parsed corpus while improving the parsing system. In Proceedings of LREC98, pp. 719-724, 1998.

[4] David McClosky, Eugene Charniak, and Mark Johnson. Effective self-training for parsing. In Proceedings of HLT-NAACL2006, pp. 152-159, 2006.

[5] Keiji Shinzato, Tomohide Shibata, Daisuke Kawahara, and Sadao Kurohashi. Tsubaki: An open search engine infrastructure for developing information access methodology. Journal of Information Processing, Vol. 20, No. 1, pp. 216-227, 2012.

[6] Jun Suzuki, Hideki Isozaki, Xavier Carreras, and Michael Collins. An empirical study of semi-supervised structured conditional models for dependency parsing. In Proceedings of EMNLP2009, pp. 551-560, 2009.

[7] 笹野遼平 , 黒橋禎夫 . 大規模格フレームを用いた識別モデルに基づく日本語ゼロ照応解析 . 情報処理学会論文誌 , Vol. 52, No. 12, pp. 3328-3337, 2011.

[8] 河原大輔 , 黒橋禎夫 . 格フレーム辞書の漸次的自動構築 . 自然

[1] 獲得した格フレームは、http://nlp.ist.i.kyoto-u.ac.jp/index.php?京都大学格フレーム から検索することができる。
[2] http://www.logos.ic.i.u-tokyo.ac.jp/gxp/

言語処理 , Vol. 12, No. 2, pp. 109-132, 2005.
[9] 柴田知秀 , 黒橋禎夫 . 文脈に依存した述語の同義関係獲得 . 情報処理学会第 199 回自然言語処理研究会 , 2010.

● TSUBAME e-Science Journal No.7
2012 年 12 月 26 日　東京工業大学 学術国際情報センター発行 ©
ISSN 2185-6028
デザイン・レイアウト：キックアンドパンチ
編集：　TSUBAME e-Science Journal 編集室
青木尊之　ピパットポンサー・ティラポン
渡邊寿雄　佐々木淳　仲川愛理
住所：　〒 152-8550 東京都目黒区大岡山 2-12-1-E2-1
電話：　03-5734-2087　FAX：03-5734-3198
E-mail：tsubame_j@sim.gsic.titech.ac.jp
URL：　http://www.gsic.titech.ac.jp/

TSUBAME

TSUBAME 共同利用サービス

**『みんなのスパコン』TSUBAME共同利用サービスは、
ピーク性能 2.4PFlops、18000CPUコア、4300GPU搭載
世界トップクラスの東工大のスパコンTSUBAME2.0を
東工大以外の皆さまにご利用いただくための枠組みです。**

課題公募する利用区分とカテゴリ

共同利用サービスには、「学術利用」、「産業利用」、「社会貢献利用」の3つの利用区分があり、さらに「成果公開」と「成果非公開」のカテゴリがあります。
ご利用をご検討の際には、下記までお問い合わせください。

TSUBAME 共同利用とは…

他大学や公的研究機関の研究者の**学術利用**［有償利用］

民間企業の方の**産業利用**［有償・無償利用］

その他の組織による社会的貢献のための**社会貢献利用**［有償利用］

共同利用にて提供する計算資源

共同利用サービスの利用区分・カテゴリ別の利用課金表を下記に示します。TSUBAME 2.0における計算機資源の割振りは口数を単位としており、1口は標準1ノード（12 CPUコア, 3GPU, 55.82GBメモリ搭載）の3000時間分（≒約4ヵ月）相当の計算機資源です。
1000 CPUコアを1.5日利用する使い方や、100 GPUを3.75日利用する使い方も可能です。

利用区分	利用者	制度や利用規定等	カテゴリ	利用課金
学術利用	他大学または研究機関等	共同利用の利用規定に基づく	成果公開	1口：100,000円
産業利用	民間企業を中心としたグループ	「先端研究施設共用促進事業」に基づく	成果公開	トライアルユース（無償利用）
				1口：100,000円
			成果非公開	1口：400,000円
社会貢献利用	非営利団体、公共団体等	共同利用の利用規定に基づく	成果公開	1口：100,000円
			成果非公開	1口：400,000円

産業利用トライアルユース制度（先端研究施設共用促進事業）

東工大のスパコンTSUBAMEを、より多くの企業の皆さまにご利用いただくため、初めてTSUBAMEをご利用いただく際に、無償にてご試用いただける制度です。

（文部科学省 先端研究施設共用促進事業による助成）

詳しくは、下記までお問い合わせください。

お問い合わせ

●東京工業大学　学術国際情報センター　共同利用推進室
●e-mail　kyoyo@gsic.titech.ac.jp　Tel. 03-5734-2085　Fax. 03-5734-3198
詳しくは http://www.gsic.titech.ac.jp/tsubame/をご覧ください。

GSIC
Global Scientific Information and Computing Center