Vol.2013-MPS-93 No.6
2013/5/23

統計的文法獲得モデルのための部分木ブロック化サンプリング法

進藤 裕之[1,a)]　松本 裕治[2]　永田 昌明[1]

概要：自然言語処理分野における統計的文法獲得では，確率文法モデルの学習に Gibbs サンプリング法が広く用いられている．しかしながら，木構造データを扱う場合には，Gibbs サンプリング法のように変数の値を一つずつ順番に更新していく方法では局所解に留まりやすく，十分に尤度の高い解を得られないという問題がある．この問題を解決するために，我々は新たな部分木のブロック化サンプリング法を提案する．本手法は，データ中に現れる共通の部分木まとめてブロック化し，ブロックに含まれる変数の同時分布からサンプリングを行う．そして，その部分木ブロック化サンプラーを従来のマルコフ連鎖モンテカルロ法と組み合わせて交互に実行することにより，目的関数の最適解を効率良く探索することができる．シンボル細分化文脈自由文法を用いて統計的文法獲得の実験を行ったところ，提案手法は既存手法よりも尤度の高い文法規則が獲得できることを確認した．

Blocked Subtree Sampler for Statistical Grammar Induction

Hiroyuki Shindo[1,a)]　Yuji Matsumoto[2]　Masaaki Nagata[1]

Abstract: Gibbs sampler is widely used for statistical grammar induction in natural language processing. However, by sampling only one variable at a time, the sampler suffers from local optimum due to the strong dependency between variables of tree structure. In this paper, we propose blocked subtree sampler to tackle this problem. Our sampler collects the same type of subtrees for each iteration and updates them simultaneously. Further, our method iterates the blocked subtree sampler and conventional Markov chain Monte Carlo (MCMC) sampler alternately to find the optimal solution efficiently. The experimental results of grammar induction show that our method achieves better performance compared with conventional methods.

1. はじめに

自然言語処理分野における文法獲得とは，日本語や英語などの文または構文木のデータから，コンピュータを用いて自動的に文法規則を獲得することである．例えば，文法モデルとして文脈自由文法を用いた場合，文法規則は S → VP NP のような深さ 1 の木構造として定義される．獲得された文法規則は，構文解析器や言語モデルとして，機械翻訳や自動要約システムなどに応用されている [3], [4]．従来より，Penn Treebank [8] などの構文木コーパスから，確率文法モデルを用いて統計的に文法規則を獲得する方法が提案されてきた [2], [11], [12]．統計的手法による文法獲得は，人手で作成されたルールによる発見的手法と比較して，言語や構文木のアノテーション仕様に大きく依存しないため様々なデータに適用できるという利点がある．

確率文法モデルの学習法として，Gibbs サンプリング法 [5] が広く用いられている．Gibbs サンプリング法の特徴は，複数の確率変数の同時確率分布から直接サンプルを生成するのではなく，変数を一つずつ順番に巡回してサンプリングを行うという点にある．そのため，文法獲得で用

[1] NTT コミュニケーション科学基礎研究所
[2] 奈良先端科学技術大学院大学
[a)] shindo.hiroyuki@lab.ntt.co.jp

いられる多くの確率文法モデルに対して単純な学習アルゴリズムを与え，汎用性が高いという利点がある．一方，確率文法モデルでは，木構造データに起因する変数間の強い相互依存性のため，変数の値を一つずつサンプリングする方法では局所解に留まりやすく，十分に尤度の高い解を得られないという問題点が指摘されている [2]．この問題に対する一般的な改善策として，複数の変数をまとめて同時にサンプリングを行うブロック化 MCMC 法が提案されている [1], [6]．しかしながら，これらの方法は，特定の文法理論や確率モデルに特化したアルゴリズムであったり，確率変数が二値であることを想定しているなど，使用する上で様々な制限があった．

上記の問題点を解決するために，本稿では統計的文法獲得のための新たなブロック化サンプリング法を提案する．我々の狙いは，Gibbs サンプリング法のような汎用性を失わずに，なるべく多くの変数を同時にサンプリングする方法を構築することである．提案手法は，既存のブロック化 MCMC 法と異なり，Gibbs サンプリング法などの一般的な MCMC 法と，ブロック化サンプリングを行うアルゴリズムとを独立に構築して，それらを交互に実行するというアプローチを取る．また，従来手法のように，どの変数をまとめてサンプリングするのかという定義を事前に与えるのではなく，適切なブロックを自動的に発見して同時にサンプリングを行う．したがって，特定の文法モデルに依存せず，多値変数も扱うことのできるブロック化サンプリングが可能となる．具体的な文法モデルとして，近年の高精度な構文解析器に使われているシンボル細分化文脈自由文法モデルに対して提案手法を適用したところ，提案手法は，Gibbs サンプリング法や従来のブロック化サンプリング法よりも尤度の高い文法規則が獲得できることを確認した．

2. 統計的手法による文法獲得

本研究では，構文木コーパスはあらかじめ与えられるものとして，構文木コーパスから文法モデルの基本木（elementary trees）を統計的に推定するという問題を考える．基本木とは，文法規則の基本単位となる部分木のことを指す．例えば，文脈自由文法では，NP → NP DT のような深さ 1 の部分木が基本木である．単純な文脈自由文法では，構文木を深さ 1 の部分木に分割すれば基本木の集合が一意に定まるが，それ以外の文法モデルでは，異なる基本木の組み合わせが同一の構文木を構成する可能性がある．そのため，確率文法モデルを用いて尤もらしい基本木の組み合わせを統計的に推定する必要がある．

2.1 確率文法モデル

構文木 t が与えられたもとでの基本木の集合 $\mathbf{e} = \{e_1, e_2, \ldots\}$ の事後確率は，ベイズの定理によって $P(\mathbf{e}|t) \propto P(t|\mathbf{e})\,P(\mathbf{e})$ と計算できる．ただし，$P(t|\mathbf{e})$ は，基本木 $\mathbf{e}$ が構成する全体の木構造が構文木 t と一致したときに 1，そうでないときは 0 の値を取る確率分布である．また，$P(\mathbf{e})$ は基本木の確率生成モデルである．提案手法は，任意の基本木の確率モデル $P(\mathbf{e})$ の学習に適用することが可能であるが，本稿では具体例として，近年の高精度な構文解析器に用いられている確率シンボル細分化文脈自由文法を取り上げる [9], [11].

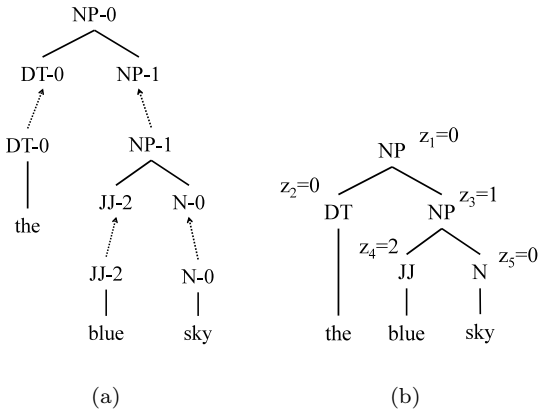


図 1 (a) シンボル細分化文脈自由文法の導出過程の例．点線の矢印は，基本木が結合する過程を表す．(b) (a) の導出過程を，構文木のノードに割り当てた変数で表現したもの．

シンボル細分化文脈自由文法は，構文木の各ノードに付与されたシンボル（非終端記号）が細分化された文脈自由文法である．シンボルを細分化することにより，例えば同じ NP（名詞句）のタグが付与されていたノードを，NP-0（文の主語になりやすい名詞句）と，NP-1（文の目的語になりやすい名詞句）のように精緻化できる．図 1（a）に，図**??**の構文木に対するシンボル細分化文脈自由文法の導出過程の例を示す．シンボル細分化文脈自由文法では，基本木 e は $A_x \to B_y C_z$ の形式を取る．ただし，A, B, C は NP や VP などのシンボルを表し，x, y, z は細分化カテゴリ（$0, 1, \ldots$）である．シンボル細分化文脈自由文法の確率モデルは，ノンパラメトリックベイズモデルの一種である Pitman-Yor 過程を用いて，以下のように定式化できる [7].

$$
\begin{aligned}
e\,|A_x &\sim G_{A_x} \\
G_{A_x} &\sim \mathrm{PYP}\left(d_{A_x}, \theta_{A_x}, P_0\left(\cdot\,|A_x\right)\right)
\end{aligned}
$$

ただし，A_x は基本木 e の根ノードのシンボルである．PYP は Pitman-Yor 過程を表し，d_{A_x}, θ_{A_x} は Pitman-Yor 過程のパラメータを表す．P_0 は基底確率分布で，基本木のバックオフ確率を与える．本研究では，基底確率を $P_0(e\,|A_x) = P_{\mathrm{MLE}}(A \to B_y C_z)$ と定義する．ただし，$A \to B_y C_z$ は，根ノードのシンボル A_x の細分化情報を取り除いた部分木であり，P_{MLE} は構文木コーパスから計算される最尤推定量を表す．

2.2 Gibbs サンプリングによる確率文法モデルの学習

一般に，構文木コーパスには文法モデルの基本木の情報

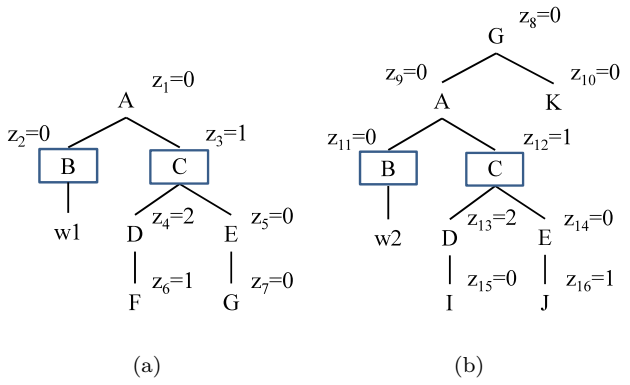


図 2　二つの構文木と，部分木のブロックの例．

は含まれていないため，構文木の各ノード（葉ノードは除く）に潜在変数 z を一つずつ割り当て，基本木の情報を表すことにする．図 1（b）に，シンボル細分化文脈自由文法の基本木の情報を潜在変数で表したものを示す．シンボル細分化文脈自由文法では，潜在変数 z の値は非終端記号の細分化カテゴリ（$0, 1, \ldots$）を表している．構文木コーパスから最適な基本木の集合を統計的に推定するには，構文木が与えられたもとでの基本木の事後確率を最大とする潜在変数とパラメータを推定すればよい．

$$\hat{\mathbf{z}}, \hat{\Theta} = \underset{\mathbf{z},\Theta}{\operatorname{argmax}}\, P\left(\mathbf{z} \,|\{t\}\,; \Theta\right) P\left(\Theta\right)$$

ただし，Θ は基本木の確率モデルのパラメータ集合である．また，推定された潜在変数 $\hat{\mathbf{z}}$ から，基本木の情報を一意に復元することができる．

事後確率を最大化する基本木を求める方法として，Gibbs サンプリング法が広く用いられている．前述のように，Gibbs サンプリング法では，基本木の同時事後分布から直接サンプルを生成するのではなく，各変数を一つずつ順番に巡回してサンプリングを行う．基本木は複数の潜在変数で構成されているため，Gibbs サンプリング法では，基本木を別の基本木へ一度に更新することはできない．したがって，ある基本木から別の基本木に至る経路中において非常に確率の低い状態が存在する場合には，基本木はいつまでも同じ状態のままに留まってしまい，最適解に到達することが困難になる．また，一つの基本木に含まれる変数をまとめて同時に更新するブロック化 Gibbs サンプリング法では，上記の問題点は解消できるが，複数の基本木を一度に更新することができないため，構文木コーパスの規模が大きくなるにつれて通常の Gibbs サンプリング法と同様の問題が生じる．また，型レベルのブロック化サンプリング法 [6] は，同じ型となる変数をまとめてブロック化し，その中でいくつの変数を反転させるかを確率的にサンプリングする方法である．同じ型の変数とは，着目している変数およびその周囲（親ノードと子ノード）の変数の値が同じである変数の集合をいう．しかしながら，型レベルのブロック化サンプリング法は，Gibbs サンプリングに完全に置き換わる MCMC 法を構成することを目的としており，事後分布に正確にしたがうマルコフ連鎖を構成するために，ディリクレ過程を事前分布として用いることや，変数の値が二値であることを仮定している[*1]．したがって，そのような制限のために，上記の Pitman-Yor 過程に基づく確率モデルなどには適用することができない．

3. 部分木ブロック化サンプリング法

3.1 部分木のブロック化

提案手法は，構文木コーパスにまたがる共通の部分木をまとめてブロック化し，それらに含まれる変数の同時分布からサンプリングを行う．まず，任意の潜在変数の集合 $\mathbf{z} = \{z\}$ に対して，それらが表す部分木を $tree\left(\mathbf{z}\right)$ とする．例えば，図 1(b) で，$\mathbf{z} = \{z_1 = 0, z_2 = 0, z_3 = 1\}$ ならば，$tree\left(\mathbf{z}\right) = \text{(NP-0 (DT-0 NP-1))}$ である．

ここで，潜在変数の集合 $\mathbf{z}$ のブロック B_s を，$B_s \equiv \{internal\left(\mathbf{z}\right) | tree\left(\mathbf{z}\right) = s \wedge \cap\mathbf{z} = \varnothing\}$ と定義する．ただし，$internal\left(\mathbf{z}\right)$ は，$\mathbf{z}$ に対応する部分木のノードの中で，非終端記号を持つ葉ノードと根ノードに相当する潜在変数を除外したものである．図 2 に，例として二つの構文木を示す．図 2 において，共通の部分木 $s = \text{(A-0 (B-0 (C-1 (D-2 E-0))))}$ に対応するブロックは，$B_s = \{\{z_2, z_3\}, \{z_{11}, z_{12}\}\}$ となる．A-0，D-2，E-0 は，部分木 s の中で非終端記号を持つ葉ノードまたは根ノードであり，これらに対応する変数の値を変更してしまうと，その周囲の基本木（例えば，(G-1 (A-0 K-0))）の情報も同時に変更してしまうことになるためブロックから除外する．B-0 は部分木 s の葉ノードであるが，前終端記号を持つので，子ノードは必ず構文木の葉ノードである．したがって，B-0 に対応する変数の値を変更しても周囲の基本木に影響を与えないため，ブロックに含める．

ブロックを構築した後に，以下の同時確率にしたがって部分木のサンプリングを行う．

$$P\left(\{\mathbf{z}\}_{\mathbf{z}\in B_s} | \mathbf{z}^{-}, \Theta\right) \tag{1}$$

ただし，$\{\mathbf{z}\}_{\mathbf{z}\in B_s}$ は，B_s に含まれる全ての変数の集合を表し，$\mathbf{z}^{-}$ は，構文木コーパス中の全ての変数から B_s に含まれる変数を取り除いた集合である．

z の値が c 通りの可能性を持つとき，式 1 の変数の値の組み合わせは，$c^{|\mathbf{z}|\times|B_s|}$ 通りである．$|B_s|$ は構文木コーパスのデータ量に応じて増大するため，全ての変数 z の値の組み合わせについて式 1 を求めることは計算量的に困難である．そこで，同じブロックに含まれる全ての変数の値は，サンプリング後も必ず同じ変数の値になるという制約を設ける．このようにすると，ブロックに含まれる変数の集合 $\mathbf{z} \in B_s$ を一つだけ取り出して，それらの値の取り得る可能性のみを考慮すればよい．し

[*1] 多値変数である場合は，サンプリングの反復毎に，スライスサンプリング [10] などの方法で二値に変換する．

たがって，組み合わせの数は $c^{|\mathbf{z}|}$ 通りになる．図 2 の例では，z が二値変数であるとすると，$(z_2, z_3, z_{11}, z_{12}) = (0,0,0,0), (0,0,0,1), \ldots (1,1,1,0), (1,1,1,1)$ の 16 通り全ての可能性について式 1 を計算するのではなく，$(0,0,0,0), (0,1,0,1), (1,0,1,0), (1,1,1,1)$ の 4 通りのみについて式 1 を計算してサンプリングを行う．上記の制約を導入して計算量を削減する代償として，式 1 に基づくサンプリング法は，基本木の事後分布にしたがうサンプルを生成しない．そこで，Gibbs サンプリング法などの一般的な MCMC 法と，上記の部分木ブロック化サンプリング法とを組み合わせて使用することを提案する．以下に，ブロック B_s の構築方法と，提案手法の具体的なアルゴリズムについて述べる．

3.2 ブロックの構築

ブロック B_s を構築するために，サンプリングの反復毎に，構文木コーパスから潜在変数を考慮した共通の部分木を探索する必要がある．我々は，パターンマイニングの手法に基づいて共通の部分木を列挙することを提案する．具体的な手順は以下の通りである．まず，一つのノードのみからなる最小の部分木から始めて，そこに子ノードを追加して部分木を拡大する．この手続きを再帰的に繰り返すと，部分木パターンをノードとする木構造が生成される．部分木の拡大は，探索中の部分木パターンがあらかじめ設定した最大ノード数に達するか，または頻度が 1 になったときに停止する部分木パターンの集合を発見した後，構文木コーパスの全ノードが必ずどこか一つのブロックに所属するまでランダムに部分木パターンを一つ選択し，そこからブロック B_s を構築する．上記の手続きを行うことによって，全てのノードが一度ずつ含まれたブロックの集合 $\mathbf{B} = \{B_s\}$ が構築できる．

3.3 提案手法のアルゴリズム

前述のように，提案手法は，一般的な MCMC 法と部分木ブロック化サンプリング法とを組み合わせて使用する．提案手法の具体的な手続きをアルゴリズム 1 に示す．アルゴリズム 1 の入力は，サンプリングの反復回数 I，構文木コーパス $\{t\}$，ブロック化サンプリングの頻度 f である．頻度 f については後述する．

アルゴリズム 1 では，まず，通常の Gibbs サンプリング法によって変数の値を更新する（行 4）．Gibbs サンプリング法以外の任意の MCMC 法を用いてもよい．次に，現在の反復値 i が，あらかじめ設定された部分木ブロック化サンプリング法の頻度 f の条件を満たすならば，ブロック化サンプリングを実行する．例えば，$f = 10$ とすると，10 回の Gibbs サンプリングを行う度に一度だけブロック化サンプリングを実行する．頻度 f を導入する理由は，部分木ブロック化サンプリングの実行に要する計算コストと，探索効率とのバランスを調整できるようにするためである．頻度 f が最適解の探索効率に与える影響は実験で評価する．部分木ブロック化サンプリング法では，前述のパターンマイニング手法によって共通の部分木を探索し（行 9），ブロック B_s を構築する．その後，ブロックの集合 $\mathbf{B}$ からランダムに一つのブロックを選択し，そのブロックに含まれる潜在変数の値の組み合わせについて式 1 を計算し，その確率にしたがってサンプルを生成する（行 21）．

Algorithm 1: 部分木ブロック化サンプリング法

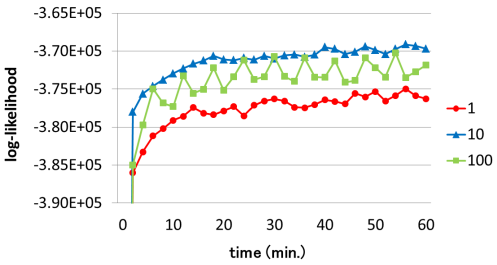
Input : number of iterations: I, parse trees: $\{t\}$, frequency of blocked subtree sampling: f
Output: estimated elementary trees:: $\hat{\mathbf{e}}$, estimated parameters: $\hat{\Theta}$

1 **for** $i = 1, \ldots, I$ **do**
2 　Initialize $\mathbf{z}, \Theta$
　 　`// Gibbs sampling`
3 　**foreach** z *in random order* **do**
4 　　Generate z' according to $P(z \,|\mathbf{z} \setminus z, \Theta)$
5 　　$z \leftarrow z'$
6 　**end**
7 　Update parameters Θ
8 　**if** $i \ mod \ f = 0$ **then**
　 　　`// Construct block`
9 　　Find subtree patterns S by subtree expansion method
10 　　$Z \leftarrow \mathbf{z}$
11 　　$\mathbf{B} \leftarrow \emptyset$
12 　　**while** $Z \neq \emptyset$ **do**
13 　　　Pick subtree $s \in S$ at random
14 　　　Construct B_s
15 　　　$\mathbf{B} \leftarrow \mathbf{B} \cup B_s$
16 　　　**foreach** z *in* B_s **do**
17 　　　　$Z \leftarrow Z \setminus z$
18 　　　**end**
19 　　**end**
　 　　`// Blocked subtree sampling`
20 　　**foreach** $B_s \in \mathbf{B}$ *in random order* **do**
21 　　　Generate $\{\mathbf{z}\}'$ according to $P\left(\{\mathbf{z}\}_{\mathbf{z} \in B_s} |\mathbf{z}^-, \Theta\right)$
22 　　　$\{\mathbf{z}\} \leftarrow \{\mathbf{z}\}'$
23 　　**end**
24 　**end**
25 **end**
26 Recover $\hat{\mathbf{e}}$ from $\hat{\mathbf{z}}$

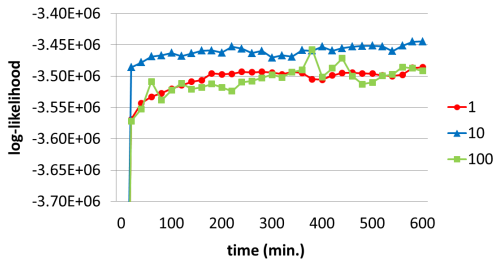
4. 実験

4.1 設定

英語の構文木コーパスである WSJ Penn Treebank [8] を用いて，提案手法の評価を行った．Penn Treebank データ

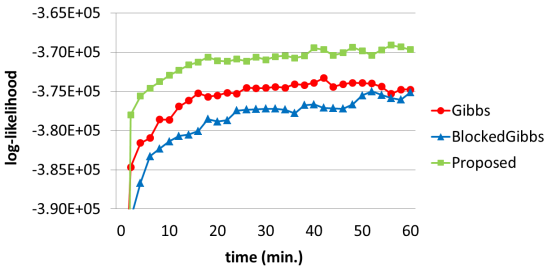


(a)

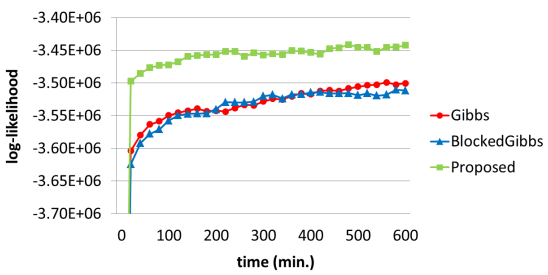


(b)

図 3　部分木ブロック化サンプリングの頻度の比較．(a) Penn-A データセットでの結果．(b) Penn-B データセットでの結果．



(a)



(b)

図 4　細分化カテゴリを 2 としたときの他手法との比較．(a) Penn-A データセットでの結果．(b) Penn-B データセットでの結果．

はセクション単位で区切られており，各セクションは約 2000 文の構文木で構成されている．本実験では，データ量の違いが各手法に与える影響を評価するため，Penn-A データ（セクション 2 のみ，1989 文）と，Penn-B データ（セクション 2 から 11 まで，18581 文）の 2 種類のデータセットを用いた．データの前処理として，コーパス中の全構文木を“Right-Binarized”法 [9] によって二分木へ変換した．また，コーパス中に 1 度しか現れない単語は，“UNKNOWN”に置き換えた．実験に用いる確率文法モデルは，2 節で説明した Pitman-Yor 過程に基づく確率シンボル細分化文脈自由文法である．サンプリングの反復数は，既存研究を参考にして 5000 回とした [13]．これは，確率文法モデルの学習結果を構文解析器で利用する際に十分な反復数である．また，実験結果で示す対数尤度は，各手法をそれぞれ独立に 5 回試行したときの平均である．使用した計算機は，CPU が Core i7 3.07GHz，メモリが 18GB である．

4.2　結果

4.2.1　部分木ブロック化サンプリングの頻度の比較

まずはじめに，提案手法のアルゴリズム 1 において，部分木ブロック化サンプリングを行う頻度 f を変化させたときの探索効率の違いを評価した．実験設定として，細分化のカテゴリ数は 2 とし，潜在変数の初期値はランダムに決定した．図 3 に，部分木ブロック化サンプリングの頻度を 1，10，100 と変化させたときの対数尤度の実験結果をグラフで示す．部分木ブロック化サンプリングの頻度によって変数の更新回数が異なるため，横軸はサンプリングの反復数ではなく，時間（分）で示してある．

図 3 に示されているように，部分木ブロック化サンプリングの頻度によって，尤度の高い解に到達するまでの時間に違いが見られた．これは，部分木ブロック化サンプリングは Gibbs サンプリングよりも計算コストが高いため，Gibbs サンプリングとブロック化サンプリングを一度ずつ交互に繰り返すよりも，10 回または 100 回ごとに実行したほうが探索効率が良い結果であったと考えられる．また，Penn-A データセットと Penn-B データセットの結果を比較すると，いずれも頻度 10 のときが最も良い結果であるが，Penn-A データセットにおいて頻度 100 は頻度 1 よりも探索効率が高いのに対して，Penn-B データセットでは，双方にあまり差が見られない．これは，データ量が増大することによって部分木ブロック化サンプリングの効果が相対的に高くなったため，小規模データの場合は，頻繁にブロック化サンプリングを行うよりも計算コストの低い Gibbs サンプリングを何度も行ったほうが探索効率が良かったのに対して，中規模データでは，計算コストをある程度要してでもブロック化サンプリングを頻繁に行ったほうが探索効率が良いという結果になったと考えられる．

4.2.2　他手法との比較

次に，提案手法と既存手法との比較実験を行った．既存手法として，Gibbs サンプリング法とブロック化 Gibbs サンプリング法を用いた．ブロック化 Gibbs サンプリング法

は，潜在変数を一つずつ巡回してサンプリングを行うのではなく，基本木を表す複数の潜在変数をまとめてサンプリングを行う方法である．例えば，図 1（b）では，潜在変数を $B_1 = \{z_1, z_2, z_3\}, B_2 = \{z_4\}, B_3 = \{z_5\}$ の 3 つのブロックに分けて，それぞれのブロックに対して式 1 を計算し，サンプリングを行う．図 4 に，提案手法と既存手法の実験結果をグラフで示す．細分化のカテゴリ数は 2 に設定し，潜在変数の初期値はランダムに決定した．また，提案手法における部分木ブロック化サンプリングの頻度は 10 に設定した．

図 4 に示されているように，提案手法は，既存手法と比較して最も探索効率が良い手法であった．Penn-A データを用いた場合，ブロック化 Gibbs サンプリング法は複数の変数をまとめて更新しているにも関わらず，通常の Gibbs サンプリング法よりも尤度の高い解に到達するのに余計に時間がかかるという結果であった．これは，小規模データでは，ブロック化された変数の同時確率を計算するために時間を多く使うよりも，少ない計算量で変数の更新回数を多くするほうが良い場合があることを示している．ただし，図 4（a）において，ブロック化 Gibbs サンプリング法と通常の Gibbs サンプリング法との対数尤度は徐々に縮まっていき，計算時間が 60 分（Gibbs サンプリング法の反復数が約 5000 回に達したとき）にはほぼ同じ値に到達した．これは，Gibbs サンプリング法が 30 分を超えた辺りから局所解に留まってしまい，それ以上尤度の高い解へ到達できない状態へ陥っているのに対し．ブロック化 Gibbs サンプリング法では，基本木を一度に別の基本木へ更新できるため，そのような問題が生じにくいためであると考えられる．一方，提案手法は，ブロック化 Gibbs サンプリング法と比較して，一つの基本木ではなく，構文木コーパスにまたがる複数の部分木をブロック単位として同時にサンプリングを行うことができるため，短時間で尤度の高い解へ到達することができる．

Penn-B データを用いた場合には，ブロッック化 Gibbs サンプリング法と通常の Gibbs サンプリング法はほぼ同じ結果であった．Penn-A データと比較すると，データ量が増大することによって，単純な Gibbs サンプリング法ではますます尤度の高い解へ到達することが困難となり，ブロック化 Gibbs サンプリング法の優位性が相対的に向上していると考えられる．提案手法は，他手法と比較して極めて良い性能である．特に，サンプリングの反復数が少なく尤度の低い状態のときに，提案手法では構文木全体にまたがる複数の変数をまとめて更新することによって尤度の高い状態へ短時間で到達していることが確認できる．

5. おわりに

本稿では，統計的文法獲得において Gibbs サンプリング法が局所最適解に留まりやすく尤度の高い解を効率的に探索できないという問題を改善するために，部分木を単位とする新たなブロック化サンプリング法を提案した．提案手法は，パターンマイニングの手法に基づいて適切なブロックを自動的に獲得し，それらをまとめて同時にサンプリングを行う．また，同じ部分木はサンプリング後も同じ変数の値になるという制約を設けて計算量を削減する代わりに，通常の MCMC 法と部分木ブロック化サンプリング法とを組み合わせて使用する．確率シンボル細分化文脈自由文法を用いて提案手法の評価を行った結果，本手法は，既存手法よりも探索効率が高く，尤度の高い文法規則が獲得できることを確認した．

参考文献

[1] Cohn, T. and Blunsom, P.: Blocked Inference in Bayesian Tree Substitution Grammars, *Proceedings of ACL*, pp. 225–230 (2010).
[2] Cohn, T., Blunsom, P. and Goldwater, S.: Inducing Tree-Substitution Grammars, *Journal of Machine Learning Research*, Vol. 11, pp. 3053–3096 (2010).
[3] Cohn, T. and Lapata, M.: Sentence Compression Beyond Word Deletion, *Proceedings of the ICCL*, pp. 137–144 (2008).
[4] Galley, M., Hopkins, M., Knight, K. and Marcu, D.: What's in a Translation Rule, *Proceedings of NAACL/HLT*, Vol. 4, pp. 273–280 (2004).
[5] Geman, S. and Geman, D.: Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-6, pp. 721–741 (1984).
[6] Liang, P., Jordan, M. I. and Klein, D.: Type-Based MCMC, *Proceedings of HLT-NAACL*, pp. 573–581 (2010).
[7] Liang, P., Petrov, S., Jordan, M. and Klein, D.: The infinite PCFG using hierarchical Dirichlet processes, *Proceedings of EMNLP-CoNLL*, pp. 688–697 (2007).
[8] Marcus, M. P., Santorini, B. and Marcinkiewicz, M. A.: Building a Large Annotated Corpus of English: The Penn Treebank, *Computational Linguistics*, Vol. 19, pp. 313–330 (1993).
[9] Matsuzaki, T., Miyao, Y. and Tsujii, J.: Probabilistic CFG with Latent Annotations, *Proceedings of ACL*, pp. 75–82 (2005).
[10] Neal, R. M.: Slice sampling, *Annals of statistics*, pp. 705–741 (2003).
[11] Petrov, S., Barrett, L., Thibaux, R. and Klein, D.: Learning Accurate, Compact, and Interpretable Tree Annotation, *Proceedings of ICCL-ACL*, pp. 433–440 (2006).
[12] Shindo, H., Fujino, A. and Nagata, M.: Insertion Operator for Bayesian Tree Substitution Grammars, *Proceedings of ACL*, pp. 206–211 (2011).
[13] Shindo, H., Miyao, Y., Fujino, A. and Nagata, M.: Bayesian Symbol-Refined Tree Substitution Grammars for Syntactic Parsing, *Proceedings of ACL*, pp. 440–448 (2012).