Vol.2016-NL-227 No.1
2016/7/29

K-best 反復ビタビパージング

林 克彦[1,a)]　永田 昌明[1,b)]

概要：本稿では確率文脈自由文法に対して，効率的，かつ，最適解の出力を保証したパージングアルゴリズムを提案する．パージングを高速化するには，探索空間から不要な辺を効率的にプルーニングすることが重要である．提案手法では探索空間を徐々に拡げながら，ビタビ内側・外側アルゴリズムを繰り返し動作させることで，最適解のモデルスコア値に対する上限と下限値を効率的に求める．そして，これらの値に基づいて辺のプルーニングを行うことで，本来の探索空間よりも大幅に縮小された空間を探索するだけで最良の導出木を発見する．本稿では英語の Penn ツリーバンクコーパスを用いた実験により，提案手法が通常の CKY 法よりも高速に動作することを示す．また，提案手法を K-best パージングに拡張する方法も示し，k-best 解を効率的に計算できることを検証する．

1. はじめに

CKY，または，ビタビ内側アルゴリズムは確率文脈自由文法 (Probabilistic context-free grammars; PCFGs) のパージングアルゴリズムとして良く知られている [7]．このアルゴリズムは動的計画法に基づいており，与えられた PCFG と入力文に対して，チャート表を使ってビタビ導出木を求めることができる．自然言語の統語解析に CKY 法は良く使われるが，文法のサイズが大きな場合，全解をしらみつぶしに解析するため，解析速度の面で大きな問題が生じる．

この CKY 法でかかるパージングの計算コストを削減するための手段として，解析中にチャート表上で生成される辺をプルーニングすることが考えられる．特に，近年開発された統語解析システムの多くでは，ビーム探索 [17] や多段探索 [1] のようなプルーニング技術が採用されており，パージングにかかる解析速度を大幅に向上させることに成功した．しかし，そのような実用面における成功の反面，前述したプルーニング技術は近似的な手法であるため，統語解析システムが最適解を出力する保証は失われる[*1]．

別の研究路線として，A*探索に基づくパージングアルゴリズムが良く研究されてきた (A*法)．A*法 [10] では元の問題よりも簡単な問題を解くことで得られる見積もり値を使って，解析中に処理する辺に優先度を付けることで解析を高速化する．また，見積もり値が *consistent* であるとき，A*法はビタビ導出木を出力することが保証される．しかしながら，文献 [19] で言及されたように，A*法は実装の観点から深刻な問題を有する:

> One of the most efficient way to implement an agenda, which keeps edges to be processed in A* parsing, is to use a priority queue, which requires a computational cost of $O(\log(n))$ at each action, where n is the number of edges in the agenda. The cost of $O(\log(n))$ makes it difficult to build a fast parser by using the A* algorithm.

そのため，著者らが知る限り，A*法に基づく実用的な統語解析システムは開発されていない．

本稿では統語解析システムが出力する解の最適性を保持しながら，不要な辺を効率的にプルーニングする新たな手法を提案する．この手法に基づくアルゴリズムを反復ビタビパージング (iterative Viterbi parsing; IVP) 法と呼ぶが，この名称の理由として，提案手法では反復的な動作がその中心的な役割を果たすからである．IVP 法は探索空間を徐々に拡げながら，ビタビ内側・外側アルゴリズムを繰り返し動作させることで，最適解のモデルスコア値に対する上限および下限値を効率的に計算する．そして，その上限，下限値を使った辺のプルーニングを行うことで，探索空間の一部を探索するだけで最適解を発見することが可能になる．IVP 法の実装は容易であり，実験から実用面でも CKY 法より高速に動作することを示す．

さらに，本稿では IVP 法を k-best 解が得られるように

[1] 日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
NTT Communication Science Laboratories, NTT Corporation
[a)] hayashi.katsuhiko@lab.ntt.co.jp
[b)] nagata.masaaki@lab.ntt.co.jp

[*1] すなわち，統語解析システムがビタビ導出木を常に出力する保証がなくなる．

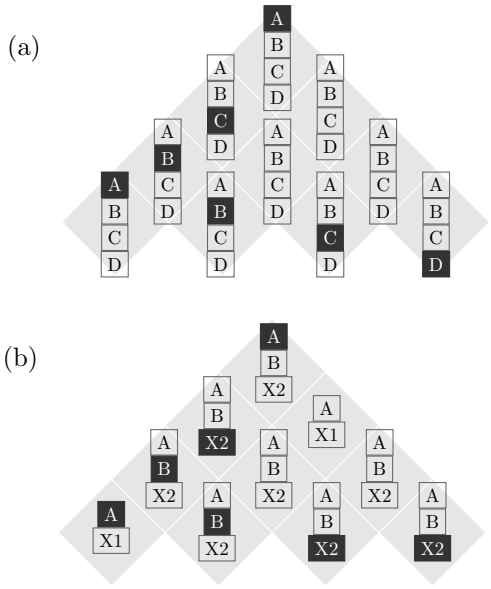

図 1 (a) 非終端記号のみから成る本来のチャート表．(b) 非終端記号及び縮退記号から成る縮退チャート表．図 (b) の黒塗りされた記号から構成される導出木 A(X2(B(A B) X2) X2) は図 (a) の黒塗りされた記号から構成される導出木 A(C(B(A B) C) D) に対応する．

拡張する方法についても示す．このアイディアは単純で，IVP 法の反復動作に Huang と Chiang の K-best アルゴリズム 3 [5] (Lazy アルゴリズム) を統合するものである．Lazy アルゴリズムはビタビ内側アルゴリズムを動作させた後，トップダウン式に k-best 解を探索する．Lazy アルゴリズムにおいて，最初のビタビ内側アルゴリズムが処理の大半を占めるため，速度面での大きなボトルネックとなる．しかし，我々の K-best IVP 法では通常の IVP 法と同じく，ビタビ内側アルゴリズムの処理を大幅に削減することが可能になる．

2. 反復ビタビパージング

文献 [14] に従って, いくつかの記法を定義する．IVP 法は PCFG G と文 $x = t_1 \dots t_n$ を入力として受け取る ($t_1 \dots t_n$ は終端記号)．一般性を失うことなく，ここでは G の文法規則がチョムスキー標準形に従うとする: 非終端記号を持つ G 中の各規則 r が A $\to$ B C の形式をとる．ここで A, B, C は非終端記号であり，この規則の対数尤度を $\log q(r)$ として書く．チャート辺，または，辺とは記号付きの入力文スパン $e = (\mathrm{A}, i, j)$ とする．辺 $e = (\mathrm{A}, i, j)$ の内側導出とは記号 A をルートとして終端記号 $t_{i+1} \dots t_j$ を葉として広がる木とする．辺 e に対する最良の内側導出のスコア $\beta(e)$ はビタビ内側スコアと呼ばれる．PCFG パージングの目的は特殊な記号 TOP を持つ終了辺 $(\mathrm{TOP}, 0, n)$ のビタビ内側スコアを求めることである．各終了辺に対して，その導出を終了導出と呼ぶ．TOP $\to t_1 \dots t_i$ A $t_{j+1} \dots t_n$ の導出に対する最良のスコア $\alpha(e)$ をビタビ外側スコアと呼ぶ．

ここで非終端記号の集合 $\{\mathrm{A}, \mathrm{B}, \mathrm{C}, \mathrm{D}\}$ を考える．仮にチャート表のセルに現れるいくつかの非終端記号をグループ化すると，本来のチャート表よりも縮退されたチャート表を構築することができる．図 1 (a) は非終端記号のみから成る本来のチャート表を表している一方，図 1 (b) のチャート表は非終端記号と縮退された記号 X1 と X2 から成る．この新たな記号はいくつかの非終端記号をグループ化することで作られ，ここでは縮退記号と呼ぶ．また，縮退記号を含むチャート表を縮退チャート表と呼ぶ．例えば，縮退記号 X1 と X2 はそれぞれ非終端記号の集合 $\{\mathrm{B}, \mathrm{C}, \mathrm{D}\}$ と $\{\mathrm{C}, \mathrm{D}\}$ とをグループ化して作られており，図 1 (b) は縮退チャート表である．

縮退記号 X, X′, X″ が現れる文法規則の対数尤度パラメータを次のように設定する:

$$
\begin{aligned}
\log q(\mathrm{X} \to \mathrm{B\ C}) &= \max_{\mathrm{A} \in I(\mathrm{X})} \log q(\mathrm{A} \to \mathrm{B\ C}) \\
\log q(\mathrm{A} \to \mathrm{X\ C}) &= \max_{\mathrm{B} \in I(\mathrm{X})} \log q(\mathrm{A} \to \mathrm{B\ C}) \\
\log q(\mathrm{A} \to \mathrm{B\ X}) &= \max_{\mathrm{C} \in I(\mathrm{X})} \log q(\mathrm{A} \to \mathrm{B\ C}) \\
\log q(\mathrm{A} \to \mathrm{X\ X'}) &= \max_{\mathrm{B} \in I(\mathrm{X}), \mathrm{C} \in I(\mathrm{X'})} \log q(\mathrm{A} \to \mathrm{B\ C}) \\
\log q(\mathrm{X} \to \mathrm{X'\ C}) &= \max_{\mathrm{A} \in I(\mathrm{X}), \mathrm{B} \in I(\mathrm{X'})} \log q(\mathrm{A} \to \mathrm{B\ C}) \\
\log q(\mathrm{X} \to \mathrm{B\ X'}) &= \max_{\mathrm{A} \in I(\mathrm{X}), \mathrm{C} \in I(\mathrm{X'})} \log q(\mathrm{A} \to \mathrm{B\ C}) \\
\log q(\mathrm{X} \to \mathrm{X'\ X''}) &= \max_{\substack{\mathrm{A} \in I(\mathrm{X}) \\ \mathrm{B} \in I(\mathrm{X'}) \\ \mathrm{C} \in I(\mathrm{X''})}} \log q(\mathrm{A} \to \mathrm{B\ C}).
\end{aligned}
$$

ここで $I(\mathrm{X})$ は縮退記号 X からそれに対応する非終端記号の集合への写像とする．この構築により，縮退チャート表上の導出スコアは本来のチャート表上でそれに対応する導出スコアの上限値を与える [10]．そして，次の定理を導くことができる:

定理 1. 縮退チャート表の中で最良のスコアを持つ終了辺 $\hat{e}$ の導出が縮退記号を一つも含まないとき，その導出は本来のチャート表における最良の導出と一致する．

証明 . 付録 A.1 にて行う． □

IVP 法のアルゴリズムは 1 つの非終端記号と 1 つの縮退記号から成る縮退チャート表を初期化して始まる．次に最良の導出を探索するため，ビタビ内側アルゴリズムを行う．もし，そのビタビ内側導出が縮退記号を 1 つも含まない場合，アルゴリズムはその結果を返して，終了する．そうでない場合，チャート表を拡大して，上記の手続きを終了するまで繰り返す．

IVP 法のアルゴリズムの疑似コードを Algorithm 1 に与える．ここでは，各イテレーションごとの処理を効率化するため，不要な辺をプルーニングしている．また，これに伴う処理を効率化するには，チャート表をどのように展開するかも重要である．よって，以下ではそれらの詳細につ

Algorithm 1 反復ビタビパージング

```
1: lb ← det(x, G) or lb ← −∞
2: chart ← init-chart(x, G)
3: for all i ∈ [1 . . . ] do
4:    t̂ ← Viterbi-inside(chart)
5:    if t̂ consists of non-terminals only then
6:       return t̂
7:    end if
8:    if lb < best(chart) then
9:       lb ← best(chart)
10:   end if
11:   expand-chart(chart, t̂, G)
12:   Viterbi-outside(chart)
13:   prune-chart(chart, lb)
14: end for
```

いて述べる．

2.1 プルーニングの手続き

ある辺 $e = (\mathrm{A}, i, j)$ に対して，e を経由する最良の終了導出のスコアを $\alpha\beta(e) = \alpha(e) + \beta(e)$ とする (e のビタビ内側・外側スコア)．このとき，仮に $lb \leq \max_{e \in \mathcal{Y}} \beta(e)$ となる下限値 lb がわかっていれば，$\alpha\beta(e) < lb$ となる辺 e はもはや不要な辺であることがわかる．ここで $\mathcal{Y}$ は本来のチャート表に含まれる全ての終了辺の集合とする．$\alpha\beta(e)$ を本来のチャート表上で計算することは非常にコストがかかる一方，縮退チャート表上ではビタビ内側・外側アルゴリズムを使って，そのスコアの上限値を効率的に求めることができる:

$$\alpha\beta(e) \leq \widehat{\alpha}(e) + \widehat{\beta}(e) = \widehat{\alpha\beta}(e).$$

ここで $\widehat{\alpha}(e)$ と $\widehat{\beta}(e)$ は縮退チャート表上の辺 e のビタビ内側スコアとビタビ外側スコアとする．もし，$\widehat{\alpha\beta}(e) < lb$ となれば，上の不等式から本来のチャート表の中で e に対応する辺がもはや不要であることがわかるので，その縮退チャート表から e を安全にプルーニングすることができる．

Algorithm 1 では決定的なチャートパージング det() を使って得られる導出のスコアを下限値 lb として設定する．しかし，パージングの効率を良くするには，よりタイトな下限値をもとめることが重要になる．そのため，best() 関数を使って，現在の縮退チャート表の中で非終端記号のみを持つ最良の導出を求める．そして，そのスコアが現在の下限値を上回るとき，lb をそのスコアで更新する．これにより多くの不要な辺を解析途中でプルーニングすることが可能になる．8-10 行目の処理は 4 行目の Viterbi-inside() 関数を少し修正するだけで，それと並行して処理できるため，実際はそのような実装にしている．

重要なこととして，このプルーニングを行う場合と行わない場合において，ある文を解析し終えるのにかかるイテレーション数は全く同じになることを注意しておく．また，各イテレーションで選ばれる導出も全く同じになる．そのため，このプルーニングは各イテレーションでかかる処理を高速化することのみに寄与する操作である．

2.2 チャート表の展開手続き

Algorithm 1 の 11 行目において，現在のチャート表を展開する方法は様々に考えられる．本稿ではまず訓練コーパス上での頻度に基づいて非終端記号の順序を決め，チャート表上の各セルでは頻度の高い記号から順に展開することを考える．そして，各イテレーションにおいて，記号を展開するセルの選択は次の「均一チャート展開法」と「最良導出チャート展開法」の 2 つの方法を考える．

2.2.1 均一チャート展開法

現在のチャート表における全てのセルの非終端記号の数を 2 倍にする．この方法では非終端記号の数を N とするとき，イテレーションの最小回数は 1 回で，最大回数は $\lceil \log_2 N \rceil + 1$ 回となる．最小回数で解析が収束するとき，各セルの記号の数は 2 個であるため，計算時間は $O(n^3)$ である．ただし，本来のチャート表上での決定的なパージングにより下限値を初期化する場合，$O(n^3 N)$ の計算時間が必要になる．一方，i 番目の反復では 2^{i-1} 個の非終端記号が各セルには存在する．そのため，

$$\sum_{i=1}^{\lceil \log_2 N \rceil + 1} n^3 8^{i-1} = n^3 \frac{1 - 8^{\lceil \log_2 N \rceil + 1}}{1 - 8} = \frac{8n^3 N^3 - n^3}{7} < \frac{8}{7} n^3 N^3$$

となり，IVP 法にかかる最悪の計算時間は $O(n^3 N^3)$ となる．これは通常の CKY 法と同じ計算時間であり，提案法が最悪の場合でも通常の CKY 法と同程度の時間で動作することがわかる．

2.2.2 最良導出チャート展開法

現在のチャート表の中で最良の導出が縮退記号を持つとき，その縮退記号が現れるセルだけ非終端記号の数を 2 倍にする．この戦略ではイテレーションの最大回数は $n^2 \cdot (\lceil \log_2 N \rceil + 1)$ となるため，理論上の理にかなった計算量を見積もることはもはや難しい．しかしながら，我々の予備実験では，均一チャート展開法よりも最良導出チャート展開法の方が実際のパージングを格段に高速化できることがわかった．そのため，本稿では均一チャート展開法は用いず，この最良導出チャート展開法を使って実験を行った．

3. 階層型 IVP 法

近年，A*探索の良質な見積もり値を効率的に計算するため，問題の階層構造を利用する手法 (階層型 A*探索) が盛んに研究されている [3]．自然言語の統語解析においても，非終端記号を階層的に定義し，各階層の記号に基づく文法

をそれぞれ構築しておくことがある (階層文法)．そして，階層文法における低次の文法を使った解析結果を高次の文法による解析の見積もり値として段階的に利用する階層型A*法の研究が行われた [13]．本稿の IVP 法でもこのような階層的な知識を利用することができれば，問題により適した探索を行うことができると考えられる．

元の文法 G が持つ非終端記号の集合を Σ とする．ここで Σ の記号を何らかの方法により $m+1$ 段階の階層的なクラスタとして定義し，各階層の記号の集合を $\Sigma_0, \ldots, \Sigma_m$ $(\Sigma_m = \Sigma)$ とする．ある $i \in [0 \ldots m-1]$ に対して，Σ_i の要素は i 階層の縮退記号と呼ぶ．また，ある $0 \leq i \leq j \leq m$ に対して，階層的な記号の定義では Σ_i の要素から Σ_j の部分集合への写像 $\pi_{i \to j} : \Sigma_i \mapsto \mathfrak{P}(\Sigma_j)$ を考える．i 階層から j 階層の記号への対応がない場合，この写像は空集合を返す．また，$i = j$ のとき，$\pi_{i \to j}$ は入力となる要素をそのままシングルトンとして返す．この写像を使って，任意の $0 \leq i, j, k \leq m$ の階層の記号に対して (i, j, k のうち，少なくとも 1 つは m より小さい)，縮退記号を含む文法規則の対数尤度パラメータは

$$\log q(\mathrm{X}_i \to \mathrm{X}_j\ \mathrm{X}_k) = \max_{\substack{\mathrm{A} \in \pi_{i \to m}(\mathrm{X}_i) \\ \mathrm{B} \in \pi_{j \to m}(\mathrm{X}_j) \\ \mathrm{C} \in \pi_{k \to m}(\mathrm{X}_k)}} \log q(\mathrm{A} \to \mathrm{B}\ \mathrm{C})$$

として求めることができる．

本稿では文献 [1] の定義を変更して，付録 A.2 のような非終端記号の階層的なクラスタを定義した．品詞タグについても文献 [16] を参考にして，付録 A.3 のように定義して実験に用いた．このような階層的な記号定義を使った IVP 法 (階層型 IVP 法) は，0 階層のラベルから成る縮退チャート表を初期化して始まり，元の非終端記号のみから成る導出が見つかるまで IVP 法と同じように反復処理を繰り返す．チャート表上で展開するセルの選択は最良導出チャート展開法と同じ基準で行い，その導出が i 階層の縮退記号を含むとき，その記号に対応する次の階層の記号集合をそのセルに展開する．

4. K-best 拡張

文献 [5] の Lazy アルゴリズムはビタビ内側アルゴリズムを動作させた後，トップダウンに k-best 導出木を探索する．K-best IVP 法は IVP 法の反復処理と Lazy アルゴリズムを統合することで，とりわけ Lazy アルゴリズムのビタビ内側アルゴリズムを高速化することに狙いがある．Alogirhtm 2 に K-best IVP 法のアルゴリズムを示す．基本的な動作は IVP 法と同じであるが，最良の導出木が非終端記号のみから成るとき，K-best IVP 法では Lazy アルゴリズムを動作させて k-best 導出木を探索する．もし，全ての k-best 導出木が縮約記号を 1 つも含まないとき，その結果を返して，アルゴリズムは終了する．

K-best IVP 法でも不要な辺をプルーニングして，各反復イテレーションごとの処理を効率化する．まず，本来のチャート表上でビーム幅 k のビーム探索に基づくパージングアルゴリズムを動かすことで，k 個の終了導出を得る．そのとき，k 番目の終了導出のスコアを下限値 lb として初期化する．プルーニングをより効率化するため，各イテレーションごとに lb を更新する．k-best() 関数を使って，現在のチャート上において，非終端記号のみから成る導出の中で k 番目に最良の導出のスコアを求め，この値が lb を上回るとき，更新を行う．k-best() 関数の具体的な手続きは，Lazy アルゴリズムを非終端記号から成る辺だけを使って動作させることで行える[*2]．

Algorithm 2 K-best 反復ビタビパージング

```
lb ← beam(x, G, k) or lb ← −∞
chart ← init-chart(x, G)
for all i ∈ [1 . . . ] do
    t̂_1 ← Viterbi-inside(chart)
    if t̂_1 consists of non-terminals only then
        [t̂_2, . . . , t̂_k] ← Lazy K-best(chart)
        if All of [t̂_2, . . . , t̂_k] consist of non-terminals only then
            return [t̂_1, t̂_2, . . . , t̂_k]
        else
            t̂_1 ← getShrinkDerivTree([t̂_2, . . . , t̂_k])
        end if
    end if
    if lb < k-best(chart, k) then
        lb ← k-best(chart, k)
    end if
    expand-chart(chart, t̂_1, G)
    Viterbi-outside(chart)
    prune-chart(chart, lb)
end for
```

他の動作原理は IVP 法と同様であり，下限値 lb が常に k 番目に最良の導出のスコアを下回るように設定されているため，このアルゴリズムの正当性は明らかである．

5. 実験

5.1 実験設定

実験では英語ペンツリーバンクの Wall street journal 部分のセクション 02–21 を訓練データ，セクション 22 の長さ 1–35 の文をテストデータとして用いた．訓練データから機能ラベル，空範疇タグ，フィラー表示を取り除いて，右分岐の 2 分化を施した後，チョムスキー標準形に従う PCFG を最尤推定で求めた．このような PCFG は比較的小規模な部類の文法ではあるが，通常の CKY 法はこれよりも大

[*2] この際に使うビタビ内側スコアは Viterbi-inside() 関数内で計算した非終端記号のみから成る導出のスコアを使っており，これは 1-best 時の下限値計算と同様の手続きである．

表 1 CKY 法，IVP 法，及び，階層型 IVP 法 (HIVP) の比較: 解析時に展開された辺の数，プルーニングされた辺の数，収束までのイテレーション数，解析時間を示している．最終行はそれぞれの結果に対する平均値を示す．

	CKY 法		IVP 法				HIVP 法			
文長	edges	time	edges	pruned edges	iters	time	edges	pruned edges	iters	time
20	10590	1.25	5290	4499	56	0.24	2864	2089	68	0.13
23	13938	1.76	5342	4447	60	0.06	2219	1462	41	0.06
22	12771	1.52	4516	3767	58	0.08	2204	1425	46	0.05
17	7701	0.72	3791	3237	42	0.05	1526	1119	32	0.03
28	20538	3.14	15801	13830	134	4.88	7306	5338	144	1.18
34	30141	5.44	15377	13150	134	2.74	6390	4634	98	0.49
⋮	⋮			⋮				⋮		
21	12801	1.77	7864	6764	74	1.71	3502	2456	70	0.21

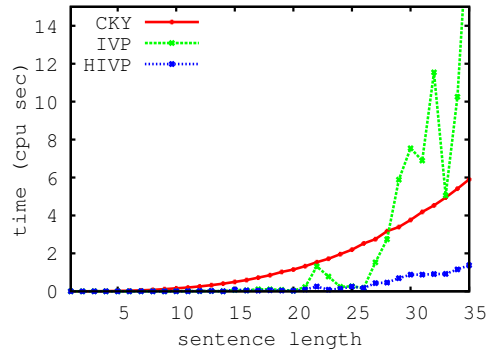

図 2 CKY 法，IVP 法，階層型 IVP 法 (HIVP) に対するテストデータの文長ごとの解析時間．

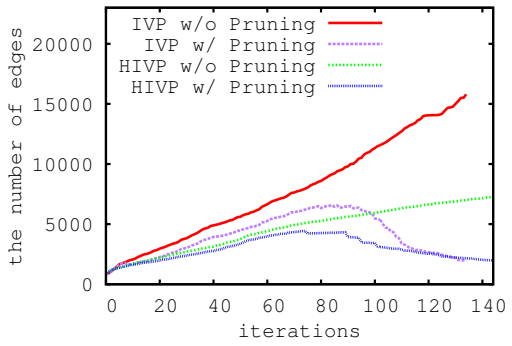

図 3 文長 28 の文に対する IVP 法，及び，階層型 IVP 法 (HIVP) における各イテレーションごとの辺の数: w/はプルーニングあり，w/o はプルーニングなしを意味する．

きなサイズ (記号数が増える) になると効率的に動作しないため，提案法の有効性を示すためのベンチマークデータとしては十分に有効である．解析時間の計測には Xeon 5600 3.33GHz の CPU マシンを利用した．

5.2 1-best パージングの実験

表 1 では CKY 法，IVP 法，階層型 IVP 法を用いてテストデータを解析したときの結果を比較する．実験からは階層型 IVP 法が CKY 法に比べて 8–9 倍程度高速に動作することがわかる．一方，通常の IVP 法は短い文では CKY 法よりも高速に動作したが，テストデータ全体での平均解析速度は同程度であった．図 2 にはテストデータの文長ごとの平均解析時間を示した．長い文に対して，IVP 法の動作が不安定である一方，階層型 IVP 法は安定して動作していることがわかる．

IVP 法と階層型 IVP 法では展開される辺の数が平均で 2 倍以上異なっている．これは階層型 IVP 法で使われる縮退記号の方が問題により適したクラスタになっているためである．すなわち，イテレーションの初期に選ばれる最良の導出が本来の最良の導出とより良く対応がとれているため，イテレーションごとに展開される縮退記号の総数が減ったと考えられる．図 3 では表 2 に示している文長 28 の文に対する IVP 法と階層型 IVP 法におけるイテレーションごとの辺の数を示した．図 3 から階層型 IVP 法では辺の数の増加を少なく抑えられていることがわかる．また，両手法ともイテレーションの中盤に辺のプルーニングが良く機能していることがわかる．

5.3 K-best パージングの実験

1-best パージングでの実験結果を踏まえて，K-best パージングの実験では Lazy アルゴリズムと K-best 階層型 IVP 法 (Kbest HIVP 法) の比較を行う．前述したように，Lazy アルゴリズムの主要な処理は最初に CKY 法を動作させる点である．K-best HIVP 法では 1-best の場合と同じく，その処理を減らすことが高速化のポイントになる．予備実験から，ビーム探索による下限値の設定は計算コストが非常に大きかったため，下限値の初期値は $-\infty$ に設定した．

図 4 では Lazy アルゴリズムと K-best HIVP 法のテストデータに対する解析時間の平均をプロットした．K-best HIVP 法は k を小さく設定したとき ($k \leq 64$ 程度)，Lazy

表 2　K-best HIVP 法の実験結果.

	$k=8$				$k=32$				$k=128$			
文長	edges	pruned edges	iters	time	edges	pruned edges	iters	time	edges	pruned edges	iters	time
20	2991	2079	75	0.33	3433	2149	103	0.72	3810	1997	144	2.74
23	2247	1160	44	0.16	2518	1215	65	0.40	3223	1321	108	1.21
22	2493	1441	63	0.13	2899	1427	100	0.69	3464	1041	146	2.37
17	1699	1169	45	0.07	1987	1096	67	0.32	2380	975	102	1.07
28	7654	5488	167	2.15	8312	5779	203	2.76	9062	5560	251	7.13
34	6390	4600	98	0.74	6577	4571	106	0.94	6996	4324	120	1.71
⋮		⋮				⋮				⋮		
21	3767	2515	84	0.52	4193	2548	108	1.01	4809	2479	145	3.92

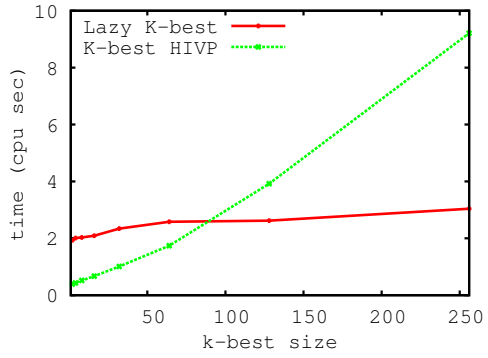

図 4　Lazy 法と K-best HIVP 法の平均解析時間の比較: k は 2, 4, 8, 16, 32, 64, 128, 256 に設定してプロットした.

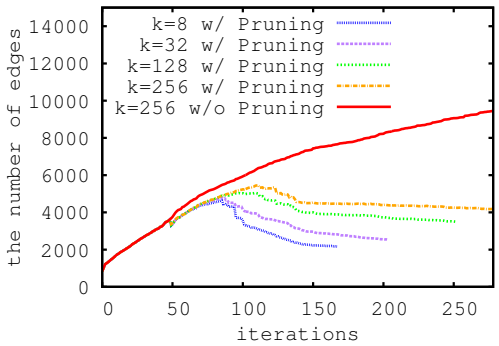

図 5　文長 28 の文に対する K-best HIVP 法の各イテレーションごとの辺の数.

アルゴリズムより高速に動作した．しかし，k を大きく設定すると，K-best HIVP 法の解析速度は大幅に減速した．表 2 には展開された辺数，プルーニングされた辺数，収束までのイテレーション数，解析時間を示した．k が大きくなる程，展開される辺数，及び，イテレーション数が増加していることがわかる．図 5 には文長 28 の文に対するイテレーションごとの辺の数を示した．k が大きな場合，展開される変数とイテレーション数の増加が進む一方，辺のプルーニングがあまり行われていないため，大幅な減速につながったと考えられる．

6. 関連文献

文献 [19] では反復 CKY 法を提案している．このアルゴリズムはある閾値を使って，それを超える導出木が見つかるまで CKY 法を反復させる点で我々の IVP 法と類似している．しかし，IVP 法ではチャート表を徐々に展開していくことで，より良い上限，下限値を効率的に計算して更新しながら，プルーニングを行う点で異なる．また，反復 CKY 法を K-best 化する方法は自明ではない．

文献 [10] では A*探索を構文解析に応用する方法が提案されている (A*法)．A*法ではオフライン，または，オンラインで本来の文法よりも簡単な文法を用いて計算できる見積もり値を利用することで解析を高速化する．文献 [13] では階層型 A*探索 [3] に基づいて，階層文法 [1], [15] を利用した A*法を提案している (階層型 A*法).

系列デコーダに対して，文献 [8] では反復ビタビ法が提案されている．また，文献 [6] ではその手法を文献 [18] の後ろ向き K-best A*探索と統合することで，k-best の系列がとれるように拡張している．本稿で提案した IVP 法はこれらのアルゴリズムをパージング問題へ一般化したものと見なすことができる．また，系列デコーダの世界では階層型 A*法や本稿の階層型 IVP 法のように記号の階層構造を利用した手法についての議論は未だ行われていない．

文献 [5] では効率的な K-best ビタビパージングアルゴリズムを提案している．本稿の K-best IVP 法は CKY 法を高速化するのと同じ要領で，この K-best アルゴリズムを高速化している．文献 [4] ではクヌースパージング法 [9], [11] を K-best パージングアルゴリズムへと拡張している．文献 [14] では階層文法を使うことで効率的に見積もり値を計算して，K-best クヌースパージング法を K-best A*法へと拡張している．

7. まとめ

本稿では PCFGs に対する効率的，かつ，解の最適性を保証するパージングアルゴリズムを提案した．このアルゴ

リズムは動的計画法に基づいて設計されているため，実装は容易に行える．今後の課題としては，潜在変数付き PCFGs [2], [12] のような非終端記号の数が大幅に増える文法を使って，IVP 法の有効性を検証したいと考えている．また，本稿の実験で示したように階層型 IVP 法が有効であることから，階層化された潜在変数付き PCFGs [15] の統語解析にそれを応用することも興味深い課題と言える．

参考文献

[1] Charniak, E., Johnson, M., Elsner, M., Austerweil, J., Ellis, D., Haxton, I., Hill, C., Shrivaths, R., Moore, J., Pozar, M. et al.: Multilevel coarse-to-fine PCFG parsing, *Proceedings of the HLT-NAACL*, pp. 168–175 (2006).
[2] Cohen, S. B., Stratos, K., Collins, M., Foster, D. P. and Ungar, L.: Spectral learning of latent-variable PCFGs, *Proceedings of the ACL*, pp. 223–231 (2012).
[3] Felzenszwalb, P. F. and McAllester, D.: The generalized A* architecture, *Journal of Artificial Intelligence Research*, Vol. 29, pp. 153–190 (2007).
[4] Huang, L.: K-best Knuth algorithm, `http://cis.upenn.edu/~lhuang3/knuth.pdf` (2005).
[5] Huang, L. and Chiang, D.: Better K-best parsing, *Proceedings of the IWPT*, pp. 53–64 (2005).
[6] Huang, Z., Chang, Y., Long, B., Crespo, J.-F., Dong, A., Keerthi, S. and Wu, S.-L.: Iterative Viterbi A* algorithm for k-best sequential decoding, *Proceedings of the ACL*, pp. 611–619 (2012).
[7] Jurafsky, D. and Martin, J. H.: *Speech and Language Processing*, Prentice Hall (2000).
[8] Kaji, N., Fujiwara, Y., Yoshinaga, N. and Kitsuregawa, M.: Efficient staggered decoding for sequence labeling, *Proceedings of the ACL*, pp. 485–494 (2010).
[9] Klein, D. and Manning, C. D.: Parsing and hypergraphs, *Proceedings of the IWPT*, pp. 123–134 (2001).
[10] Klein, D. and Manning, C. D.: A* parsing: Fast exact Viterbi parse selection, *Proceedings of the HLT-NAACL*, pp. 119–126 (2003).
[11] Knuth, D. E.: A generalization of Dijkstra's algorithm, *Information Processing Letters*, Vol. 6, No. 1, pp. 1–5 (1977).
[12] Matsuzaki, T., Miyao, Y. and Tsujii, J.: Probabilistic CFG with latent annotations, *Proceedings of the ACL*, pp. 75–82 (2005).
[13] Pauls, A. and Klein, D.: Hierarchical search for parsing, *Proceedings of the HLT-NAACL*, pp. 557–565 (2009).
[14] Pauls, A. and Klein, D.: K-best A* parsing, *Proceedings of the ACL*, pp. 958–966 (2009).
[15] Petrov, S., Barrett, L., Thibaux, R. and Klein, D.: Learning accurate, compact, and interpretable tree annotation, *Proceedings of the COLING-ACL*, pp. 433–440 (2006).
[16] Petrov, S., Das, D. and McDonald, R.: A universal part-of-speech tagset, *arXiv preprint arXiv:1104.2086* (2011).
[17] Ratnaparkhi, A.: Learning to parse natural language with maximum entropy models, *Machine Learning*, Vol. 34, No. 1-3, pp. 151–175 (1999).
[18] Soong, F. K. and Huang, E.-F.: A tree-trellis based fast search for finding the n-best sentence hypotheses in continuous speech recognition, *Proceedings of the ICASSP*, pp. 705–708 (1991).
[19] Tsuruoka, Y. and Tsujii, J.: Iterative CKY parsing for probabilistic Context-free Grammars, *Proceedings of the IJCNLP* (2006).

付　　録

A.1　定理 1 の証明

証明． $\mathcal{Y}$ を本来のチャート表に含まれる全ての終了辺の集合とする．$\mathcal{Y}' \subset \mathcal{Y}$ を $\mathcal{Y}$ の部分集合とし，この集合の要素は縮退チャート表には現れない全ての終了辺とする．$\mathcal{Y}''$ は縮退チャート表に現れる全ての終了辺の集合とする．ここで各終了辺 $e \in \mathcal{Y}'$ に対して，縮退チャート表にはそれに対応する終了辺の導出が唯一存在する．このことは図 1 に示した導出木から明らかである．このとき，

$$\forall e \in \mathcal{Y}, \exists e' \in \mathcal{Y}'',\ \beta(e) \leq \beta(e') < \beta(\hat{e})$$

は明らかであり，これは $\hat{e}$ が本来のチャート表においても最良の終了辺であることを意味する． □

A.2　非終端記号の階層的簡略化

階層	0	1	2
HP		S_	S
			VP
			UCP
			SQ
			SBAR
			SBARQ
			SINV
		N_	NP
			NAC
			NX
			LST
			X
			UCP
			FRAG
MP		A_	ADJP
			QP
			CONJP
			ADVP
			INTJ
			PRN
			PRT
		P_	PP
			PRT
			RRC
			WHADJP
			WHADVP
			WHNP
			WHPP

A.3 品詞タグの階層的簡略化

階層 0	1	2
Op_	ADJ_	JJ
		JJR
		JJS
	ADV_	RB
		RBR
		RBS
		WRB
	NOUN_	NN
		NNP
		NNPS
		NNS
	VERB_	MD
		VB
		VBD
		VBG
		VBN
		VBP
		VBZ
Cl_		IN
		CC
		CD
	DET_	DT
		EX
		PDT
		WDT
	PRON_	PRP
		PRP$
		WP
		WP$
	PRT_	POS
		RP
		TO
Ot_	X_	FW
		LS
		SYM
		UH
	PUNC_	#
		$
		”
		“
		-LRB-
		-RRB-
		,
		:
		.