自 然 言 語 処 理 74 － 2
( 1 9 8 9. 9. 2 1 )

# 日本語解析における最適解探索

平川秀樹　天野真家

(株)東芝　総合研究所

自然言語文の解析における最大の問題点は、形態素／構文／意味／文脈などの各種レベルにおける曖昧性の中から、いかにして正しい解釈を認識するかという点である。各レベルにおいて組み合わせ的な解釈が存在するばかりでなく、一般に、各種レベルの知識は、a.ある解釈の優先性(preference)に関する場合が多い、(その知識を制約的(restrictive) に適用できない)、b. 異なったレベルにおける知識の干渉が多々あるなどの性質をもっている。このため、組み合せ爆発を避けるために決定的な処理を導入することはシステムの解析能力を限定してしまう。本論文では、日本語の構文／意味解析処理において、意味係り受けグラフというデータ構造を導入することにより、文の意味解釈に関する曖昧性を内在して表現し、その中から、構文・意味・ヒューリスティックス知識により設定された優先度が最大である意味解析木(最適解)を探索する手法について述べる。最適解の探索には、分岐限定法(Branch-and-bounding method)を用い、実験によりそのパフォーマンスを検討・評価する。

# Method for Searching Optimum Tree in Japanese Sentence Analysis

Hideki Hirakawa　Shin-ya Amano

Toshiba Corp. Research and Development Center
Information systems lab.
1,Komukai Toshiba-cho,Saiwaiku,Kawasaki,210,Japan

There are combinatorial ambiguities in each level of natural language analysis, such as morphology, syntax, semantic level. Moreover, much of linguistic knowledge in each level is preference knowledge and has mutual interference. Deterministic processing is usually introduced to avoid the combinatorial explosion. However, this will restrict the ability of natural language processing system because of the above-mentioned features of linguistic knowledge. This paper describes a sentence analysis method which uniformly evaluates syntactic and semantic preference knowledge, and shows an algorithm (based on Branch-and-bounding method) to search the optimum semantic tree from a semantic graph which holds syntactic and semantic ambiguities in a Japanese sentence.

## 1. はじめに

自然言語文の解析における最大の問題点は、形態素／構文／意味／文脈などの各種レベルにおける曖昧性の中から、いかにして正しい解釈を認識するかという点である。形態素・構文といった各レベルにおいて組み合わせ的な解釈が存在するばかりでなく、一般に、各種レベルの知識は次の性質を持っている。

a. ある解釈の優先性(preference)に関する場合が多い
(その知識を制約的(restrictive) に適用できない)

b. 異なったレベルにおける知識の干渉が多々ある

処理時間／処理空間の組み合わせ的爆発を避けるために、決定的な処理を導入する事は、現状ではシステム効率上やむをえない面もあるが、その過度の適用は、上記のような知識の性質を無視することになり、システムの解析能力を限定することになるため、避けることが望ましい。このような観点から、文解析における曖昧性を増進的に解消するモデルが提唱されている[1][2]。上記a,bの性質をうまくシステムに反映するためには、

a.各種レベルの多数の解釈を効率良く保持し、

b.それらに対し各種知識による優先度を設定し、

c.その中から最適な解釈を探索する

という3つの処理が必要である[3]。aに関しては、例えば構文木を効率良く保存するために、それらをパックして保存する方法([4],[5],[6]) が提案されている。また、制約依存文法による解析では、制約マトリックスを用いて、係り受けの可能な解釈を内包的に保持することにより、個々の解釈に展開することなしに制約伝搬による解の絞り込みを行うワク組みを提供している[7]。我々は、[3] において、日本語の構文・意味解析に関して、上記a～cの処理を実現するため、意味係り受けネットワークというデータ構造を提案した(本論文では、都合上、意味係り受けグラフと呼ぶ)。cに関しては、組み合わせ的な計算が最終的に要求され、それをいかに効率良く実現するか、また、実際の計算機で取り扱う事が可能かといった点が問題となる。本論文では、分岐限定法(branch-and-bound method)[8]を、cの処理、すなわち意味係り受けグラフからの最適解探索に適用し、それを用いた解析実験について報告する。2章では、係り受けグラフとその構成法、3章では、最適解探索アルゴリズム、4章では、その実験結果、5章では、アルゴリズムの改良等について述べる。

## 2 意味係り受けグラフ

### 2.1 意味係り受けグラフ

日本語解析における曖昧性には、語の役割に関する曖昧性、語の係り先に関する曖昧性、語の係り受け意味関係に関する曖昧性、さらに文脈要素を考慮すれば語の指示、同一性に関する曖昧性がある。ここでは、これらのうち、語の係り先およびその意味関係に関する曖昧性に焦点をあてることとする。

語の係り先およびその意味関係を表現する手段として、意味係り受けグラフというデータ表現を用いる。このグラフにおいて、ノードは、1つの単語(文節)における1つの役割に相当する。2つのノードを結合するアークは、その2つのノード間に係り受け関係が存在することを示し、そのアークの名前は、その係り受けの意味関係を示すものとする。また、各アークには、重みと呼ばれる優先得点が付加されている。図1は、「私はあなたが好きです。」という文に対応する意味係り受けグラフである。図中のha,gaは、それぞれ"行為者","対象"に相当する意味関係である。意味係り受けグラフは、日本語の係り受け関係が、文の先頭方向から文末方向へ向かうという性質上、DAG (Directed Acyclic Graph)となる。日本語の文の意味構造は、この係り受けグラフ上のwell-formed な木である。図1の例では、4つの木が含まれているが、このうち「私-ha-> 好き<-ga- あなた(I love you)」と「あなた-ha-> 好き<-ga- 私 (You love me)」の2つの木が、well-formed な木となる。ここでwell-formed とは、次の2つの条件を満足する木である。

a. どの2つのアークを取っても、それらは交差しない(非交差条件)

b. どの2つのアークを取っても、それらは用言の同じ格を埋めない(多重格禁止条件)

図1の例では、bの条件により、well-formed な木は2つに限定されるわけである。

### 2.2 意味係り受けグラフの生成

ここで対象とする処理は、構文／意味解析処理である。このため、入力は、文節の列である。意味係り受けグラフは、a.構文解析,b.係り受け候補生成の2つの過程より生成される。図2に、意味係り受けグラフの生成を含む解析処理の過程を示す。

#### 2.2.1 構文解析

構文解析は、文節の列を入力し、曖昧性内蔵型の中間的な木構造(ここでは、G木と呼ぶ)を1つ出力する。解析には、文脈自由文法ベースのパーサを用いている。この木構造の例を図2.1aに示す。この木構造は、次の性質を持っている。

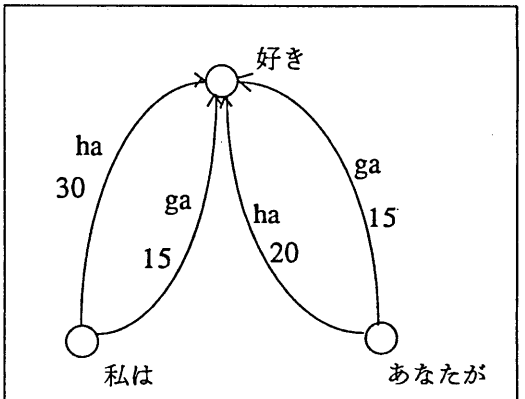



図1 「私はあなたが好きです。」の意味係り受けグラフ

a. 各ノードは入力文に現れる文節又は単語に対応する。

b. 各アークは文節間の構文的係り受け関係を表現する。

c. 各ノードの可能な係り先ノード(文節または単語)は、そのノードの祖先ノードのいずれかである。

この構造は、cの性質により、基本的には制約依存文法と同様に、可能な係り受け関係を内包させることができる。実際、最右文節を根とし、最左文節を葉とする直線状のG木は、各要素はその右側の要素に係るという制約のみが適用された状態の木である。G木の特徴は、次のようである。

a. 構文的に不可能な係り受けの可能性が排除されている。

b. 木構造自体が構文的な情報を表現している。

構文解析のフェイズでは、文脈自由文法の知識による解の絞り込みが行なわれる。

## 2.2 係り受け候補生成

2.1で得られた解析木は、明示的には全ての解釈を表現していない。また、その表現する構造は、文の構文的な情報を表現している。係り受け候補生成過程では、構文解析過程の出力(G木)を入力として、それに対する意味係り受けグラフを生成する。

各アークは、G木の中の、係り受け可能な2つのノードを取り出し、それらの間の意味係り受け関係を検索することにより、設定される。例えば、2.1で挙げた「私はあなたが好きです。」において、「私は」と「好きです」の間には、gaアーク(行為者格)とwoアーク(対象格)の2つのアークが設定される。重みは、2つのノードの係り受けの確信度を計算する知識により決定される。この知識は、構文的な情報、意味的な情報、ヒューリスティックな情報等を含ませることが可能である。例えば、次のような知識が含まれている。

・「は」が導く文節は、文末の述語に係り易い。

・係り文節と述語格スロットの意味マーカとが一致したらその係り受けの可能性が高い。

・読点があると最も近い係り要素に係りにくい。

係り受けに関して制約的に適用できる知識は、このレベルでアークの接続を禁止する事により実現できる。以上により、意味係り受けグラフがG木より得られる(図2-2c)。

## 3.最適解探索

意味係り受けグラフから重みが最大であるwell-formedな木を見つけ出す問題は、任意のDAG:G=(N,A)(Nはノードの集合、Aはアークの集合)において、制約条件:R={Ai,Aj},$i \leqq |N|-1, j \leqq |N|-1, i \neq j$ を満足する最大木を探索する問題であり、NP問題である。ここでは、既に述べたように、探索の方法として分岐限定法(branch-and-bound method)[8]を採用することとする。

## 3.1 分岐限定法

分岐限定法は、NP完全問題のような難しい問題を解く場合に用いられる計算原理である。基本的には、与えられた問題(P)を幾つかの小規模な問題に分解し、それらを解くことにより元の問題を解くものである。この計算途中で出現する部分問題について、a.何等かの方法で部分問題の最適解が求まる、あるいはb.何等かの方法で部分問題が元の問題の最適解を与えないことがわかれば、その部分問題をさらに分解する必要はない。この操作を限定操作(bounding operation)と呼び、これにより探索における枝刈りが行われるわけである。限定操作には、代表的には、上界値による限定操作と優越関係による限定操作がある。(意味係り受けグラフにおける最適解は、重さが最大である木であるので、最大値を持つように解を最適化するという条件で説明する。)これらを簡単に説明すると、次のようになる。

上界値による限定操作:

部分問題Pの条件を緩和して、より簡単な問題P′を、次の条件を満足するように作成する。

a. $g(P') \geqq f(P)$ g(P′)はP′の最適値、f(P)はPの最適値

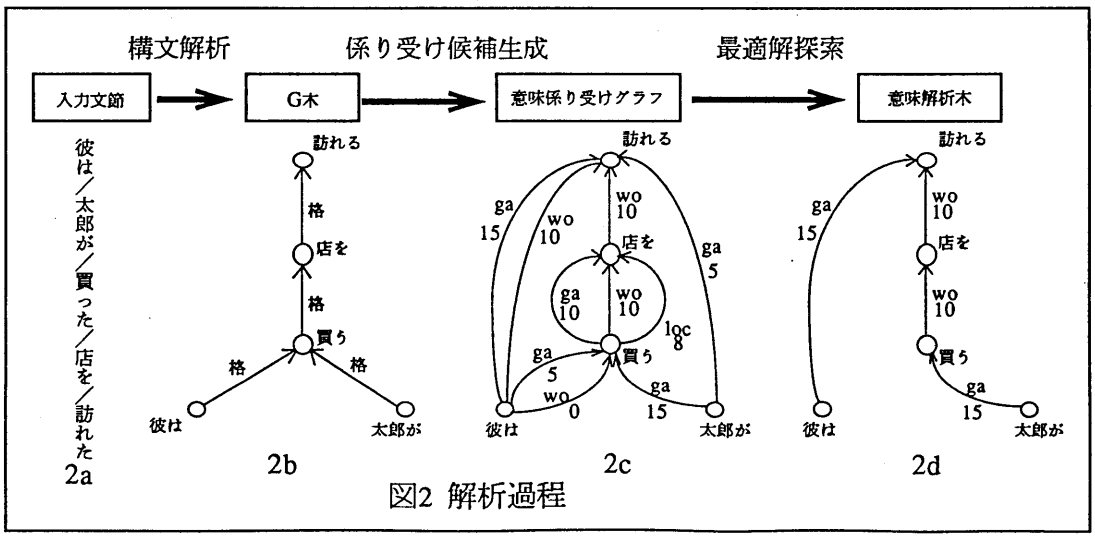



図2 解析過程

b. g(P')=f(P)ならば、P'の最適解はPの最適解

c. P'が許容解(制約条件を満たす解)を持たなければ、Pも許容解を持たない

d. すでに値zの(許容)解が他の部分問題から得られているとき、g(P')≦zならば、Pから生成される部分問題は、必ずz以下の値を持つ

ここで、b～dの場合には、部分問題Pを終端(terminate)して良い。

優越関係による限定操作:

ある問題Pを部分問題に分割した時に、その部分問題Pi,Pjの間に、f( Pi )≦f(Pj )という関係が成立するときに、Pi はPj に優越する(dominate)といい、この時、Pj を終端して良い。

ここで、図3に1つの最適解を求める分岐限定法の一般的アルゴリズムを引用する(なお、最適解を全て求めるアルゴリズムもほぼ同じフローにより実現できる)[8]。

3.2 分岐限定法の意味係り受けグラフ探索への適用

分岐限定法の各部分処理に、次に示す方法を適用することにより意味係り受けグラフG=(N,A) (Nはノードの集合、Aはアークの集合) から最大重みを持つwell-formed な木を探索するアルゴリズムを構成する。以下では、well-formed な木の条件、すなわち、任意の2つのアーク間の非交差条件および多重格禁止条件を共起条件と呼ぶ。

図3 分岐限定法アルゴリズム(P0の最適解を一つ求める)

N : すでに生成されている部分問題の集合
A : 探索図において未終端な部分問題の集合
l : l(P)が部分問題P の下界値をあたえる関数
g : g(P)が部分問題P の上界値をあたえる関数
s : s(A) は、Aから1つの部分問題を選択する関数
G : g(P)が許容解をもたないあるいはg(P)=f(P) である部分問題の集合
f : f(P)がP の最適解を現す
D : Pi D Pj ならば、PiとPjの優越関係が成り立つという関係

S1(初期値設定) : A:={P0}, N:={P0},z=-∞, O:={}
S2(探索) : A={} ならS9へ、A≠{}ならPi=s(A) としS3へ進む
S3(暫定値改良) : l(Pi)>z ならz:=l(Pi), O:={x} (xはf(x)≧l(x)を満たすPiの許容解)、S4へ進む
S4(Gテスト) : Pi Gならば S8へ、さもなければS6へ進む
S5(上界値テスト) : g(Pi) ≦z ならばS8へ、さもなければS6へ進む
S6(優越テスト) : Pk D Pi を満たすPk (≠Pi) Nが存在すればS8へ、さもなければS7へ進む
S7(分岐操作) : Piの子問題Pi1,Pi2,..Pik を生成し、A:=AU{Pi1,Pi2,..Pik}-{Pi}, N:=NU{Pi1,Pi2,..Pik}とする。S2へ戻る。
S8(Piの終端) : A:=A-{Pi} としてS2へ戻る
S9(停止) : 計算終了。計算終了時にz は、最適値f(P0)に等しく、z> -∞の時にOに記憶されているx はP0の最適解である。z= -∞ならば、P0は許容解を持たない。

・許容解を求めるアルゴリズムl(P):

アルゴリズムを図4に示す。このアルゴリズムは、基本的にはアークの重さの高い順に、共起条件を満たすように深さ優先に意味係り受けグラフを探索し許容解を1つ求める。

このアルゴリズムは、step5 において、共起条件を満足しないアークが見つかった時点でそれを解消する最も近いチョイスポイントまでバックトラックするというオブティマイゼーションを施した深さ優先探索であり、このアルゴリズムが解を持たない場合は、問題Pが解を持たない。また、このアルゴリズムの解は、問題Pの許容解である(すなわち共起制約を満たす木)であることは明白であろう。step1 において、重さの大きい順にアークグループを並び換える操作は、本質的には不必要ではあるが、できるだけ重さの大きいアークから選択してゆくことにより、出来るだけ良い(重い)木を求めるためのものである。この暫定解が良ければ、限定操作が有効に働き、余分な部分問題の生成を減少させることになる。このアルゴリズムの計算時間は、ノード数nに対して、step1 に対してソーティングO(nlog(n))、およびstep2 ～6 に対して指数関数オーダーとなるが、計算途中に共起制約によるバックトラックが無い場合には、step2 ～6 に対してO(n2)となる。このバックトラックの回数は、対象となるグラフの性質により、すなわちどの様な重み付けがされているかにより変化するものと考えられる。実際の文において、どの程度のパフォーマンスとなるかについては、後の実験により検証する。

図4 許容解を求めるアルゴリズム l(Pi)

step1 : グラフG=(N,A)(ここで|N |=n) のアークをその始点別にグループ化して、アークの集合s1,s2,..., sn-1を作る。各Siの要素を重さの大きい順に並び換える。さらに、s1,s2,...,sn-1をその中のアークの最大重さの大きい順に並び換える。これを新たに、s1,s2,...,sn-1とする。

step2 : EP=[],BP=[],i=1,j=1,w=-∞ とする。

step3 : i=n であれば、終了する。この時のEPが解である。EPの各要素aiの重みを合計しそれをw に設定する。

step4 : Siの要素数≦j であれば、step5 へ、さもなければEP=[] として終了する。(解が存在しない)

step5 : Siの j番目の要素a(i,j)と、EPの要素e1,e2,...,ei-1に関して共起条件をEPの終りからチェックし、EPのすべての要素に対して条件を満足していれば、step6 へ進む。EPの要素ek (1 ≦k ≦i-1)とa(i,j)が条件を満足しなければ、EPからek,ek+1, ...,ei-1を除き、j:=BP[K]+1,i:=k とし step4へ戻る。

step6 :EPの最後にa(i,j)を追加し BP[i]:=j,i:=i+1,j=1 とし、step3 へ戻る。

・上界値を求めるアルゴリズムg(P):

このアルゴリズムは、意味係り受けグラフの最大木を求めるものであり、基本的にアークの重さの高い順に、深さ優先に探索し最大木を求める。このアルゴリズムは、許容解を求めるアルゴリズムのstep5 における共起条件チェックを無くせばそのまま実現できる。ここで対象となっているグラフはDAGであるため、実際、各ノードから出ているアークのうち最大重みのものを集めれば、それが最大木となっていることは明白である。このため、step2 ～6 に対してO(n)で計算できる。また、最大木の重さg(Pi) は、明らかに最適木の重さf(Pi) に対して、g(Pi) ≧f(Pi) である。また、g(Pi) が解を持たないときは、f(Pi) も解を持たない。このため、計算途中で暫定値z が、z≧g(Pi) を満たした場合は、その部分問題Piは、終端することができる。

・優越テストアルゴリズムD:

本アルゴリズムでは、優越テストは用いない。

・分岐操作:

分岐操作は、部分問題Pi(グラフG(N,Ei) )の最大木を構成するアークのうちで、共起条件を満足しないアークe1,e2 を求め、新たなグラフ、Gi1=(N,Ei-{e1}),Gi2=(N,Ei-{e2}) を作成し、それに対して部分問題Pi1,Pi2 を作成するという処理とする。Piの解は、共起条件によりe1とe2の2つのアークを含まないため、この操作により得られたPi1,Pi2 のいずれかがPiの最適解を持つことになる。なお、最大木よりアークe1,e2 を求めるには、許容解を求めるアルゴリズムのstep5 において、最初に見付かった共起条件を満足しないアークのペアを記憶しておけば良い。これにより、最も重さの大きい共起条件を満足しないアークのペアe1,e2 を得ることができる。

・探索s(A):

活性部分問題の集合Aより、次に展開すべき問題を選択する。探索s(A)については、深さ優先探索、幅優先探索、最良上界探索などがあるが、ここでは、最良上界探索を採用している。すなわち、活性部分問題集合のうち最大の上界値を持つ部分問題をs(A)は選択する。この探索法は、優越テストを用いない場合において、最終的に解が得られるまでに分解される部分問題数が最小であることが知られている。以上の設定による、意味係り受けグラフの最適解探索アルゴリズムは、図5のようになる。

### 3.3 最適解探索例

本節では、例を挙げて、3.2で述べたアルゴリズムの動作例を示す。例文は、「私も彼が机を買った店に売った。」である。この文に対応する意味係り受けグラフは、図6であるとする。また、図においてa～lのアルファベットはアークのアイデンティファイアであり、ga、woなどは意味係り受けラベルであり、数値はアークの重みである。また、ここでは、多重格条件に関連する格フレームとして次を想定している。

得る ga[行為者] wo[対象] ni[相手]
買う ga[行為者] wo[対象] de[場所]

図において、「私」からは、f,g,h,iの4本のアークが出ているが、これは、副助詞「も」は、「売る」「買う」のそれぞれに対して、ga(行為者),wo(対象)の2つの格要素として係り得る可能性があることを示している。同様に、「買った」が導く埋め込み文においては、「店が買った」、「店を買った」、「店で買った」の3つの解釈が存在している。各アークの重みは、2章で述べたプロセスにより、主として単語間の係り受け評価により設定されている。

アルゴリズムの動作の説明の前に、計算途中で現れる部分問題の構成について説明する。部分問題は次の要素を持つ必

図5 意味係り受けグラフの最適解探索アルゴリズム
(P0の最適解を一つ求める)

S1(初期値設定) : A:={P0},z=-∞, O:={}
S2(探索) : A={} ならS9へ、A≠{}ならPi=s(A) としS3へ進む
S3(暫定値改良) : l(Pi)>z ならz:=l(Pi), O:={x}(xはPiの許容解)、S4へ進む。
S4(Gテスト) : g(Pi)= -∞または、g(Pi)=l(Pi) ならばS8へ、さもなければS5へ進む
S5(上界値テスト): g(Pi) ≦z ならばS7へ、さもなければS6へ進む
S6(分岐操作) : Piの子問題Pi1,Pi2 を生成し、A:=AU{Pi1,Pi2}-{Pi} とする。S2へ戻る。
S7(Piの終端) : A:=A-{Pi} としてS2へ戻る
S8(停止) : 計算終了。計算終了時にzは、最適値f(P0) に等しく、z> -∞の時にOに記憶されているx はP0の最適解である。z= -∞ならば、P0は許容解を持たない。

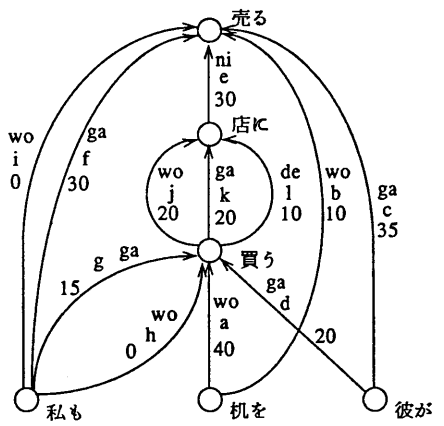



図6 「私も彼が机を買った店に売った。」に対する意味係り受けグラフ

要がある。

a. 部分問題Piのグラフ Gi
b. Piに対する許容解の値l(Pi)
c. Piに対する上界値g(Pi)

部分問題のグラフGiを各問題毎に持つのは非効率であるので、ここでは、グラフから排除されたアークのリスト(rem[] で表現する)を用いて部分問題を表現する。このため、3.2で示したアルゴリズムとは少し異なったアルゴリズムを用いることとする。すなわち、3.2の許容解を求めるアルゴリズムl(Pi) のstep1 における並び換えは、探索の初期問題にたいして一回だけ行う。図6のグラフに対しては、その結果は次のようになる。

S1[机]　a[wo,買う,40] b[wo,売る,10]
S2[彼]　c[ga,売る,35] d[ga,買う,20]
S3[店]　e[ni,売る,30]
S4[私]　f[ga,売る,30] g[ga,買う,15] h[wo,買う,0]
　　i[wo,売る,0]
S5[買う] j[wo,店,20] k[ga,店,20] l[de,店,10]

分岐された部分問題は、上記のデータからrem[...]で表現されたアークを除いたグラフを対象として持つ。
図7に、上記例に対する計算過程を示す探索図(search diagram) を示す。図において、ノードは部分問題を示し、それぞれ許容解値l、上界値g、グラフrem 、および共起条件を満たさないアークのペアcを持つ。zは、その部分問題を計算する時点での暫定解の値を示す。また、Pi のi は問題が分岐された順序を示している。図のP0 は初期問題であり、rem[] である。この問題の許容解は、図に示すように(a,c,e,h,k) であり、その許容解値lは125 である。また、上界値gは155 である。ここでは、3.2のアルゴリズムに従い、最大木中に含まれる、共起条件を満足しないアークペアc=(c,f) (多重格禁止条件)が求められている。暫定解値zは、その初期値－∞であるため、許容解値l=125がzに設定される。c=(c,f)により、P0 は、2つの部分問題P1,P2 に分岐される。P1 とP2 のl,g,cがそれぞれ計算され、図に示すような結果となる。ここで、次に処理されるべき問題を検索する。ここでは、P1,P2 が対象であるが、共にg=140であるので、いずれが選択されても良い。P1 が選ばれたとする。以下同様に計算が進行する。P3 では、rem[c,a]となり、このグラフに対しては、許容解が存在しなくなり終端する。また、P7 では、上界値gが125 であり、暫定解値zが130 であるため、限定操作により終端される。以上の処理により、図8に示す意味木が最適解として得られる。

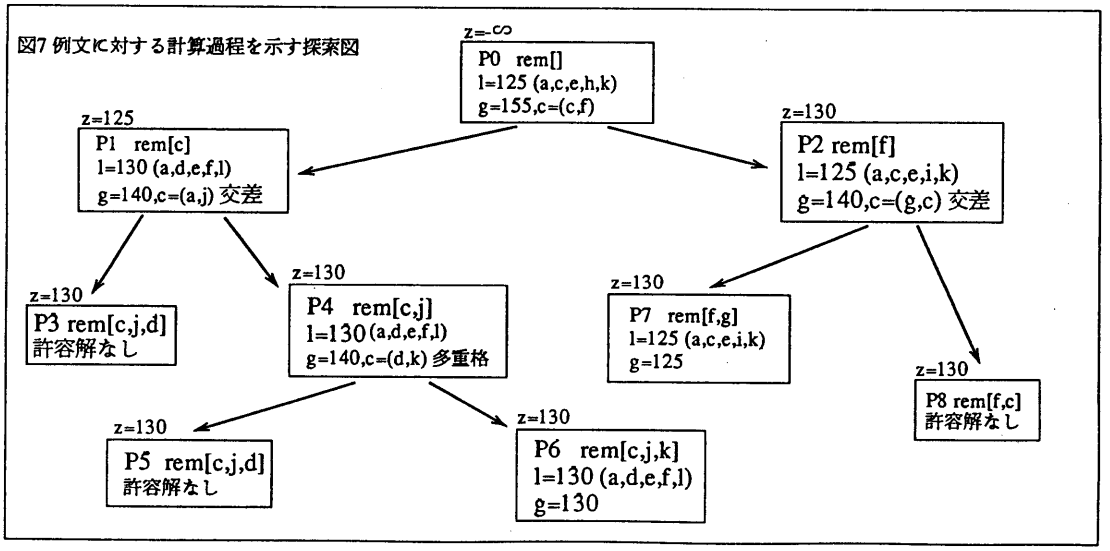
### 3.4 アルゴリズムの計算量

分岐限定法の計算の効率を評価する上で重要なパラメータは、生成される部分問題の個数Tである。一般に、Tは、分岐図の高さhに対して指数オーダーで増加するため、上記のアルゴリズムの生成する部分問題数は、意味係り受けグラフのノードの数、すなわち入力文の長さに対して、基本的に指数オーダーの部分問題を生成する。Tの下界は、部分問題Piの上界値g(Pi) と最適解値f(Pi) に関して次の関係が成立することが知られている。

$$T \geqq |\{Pi \mid g(Pi) > f(P0)\}|$$

このため、g(Pi) の精度が良ければ、それだけ最低限生成される部分問題数が低くなる。上記アルゴリズムでは、g として最大木を使用しているが、これにより実際の文(通常、最大で40程度のノード数を持つ)の解析において、どの程度の部分問題数に抑えられるかが、使用上の1つのポイントとなる。また、計算に必要な記憶スペースには、主として、a.意味係り受けグラフ、b.部分問題の2つがある。意味係り受けグラフは、2つのノード間にk 個の意味係り受け関係が存在

図7 例文に対する計算過程を示す探索図



するとすれば、最大kn(n-1)/2 本のアークを有するので、O(n2) のスペースを要する。また、部分問題のための記憶スペースは、(部分問題1つの記憶スペース)×(計算途中の分岐図における葉の最大数)となる。1つの部分問題には、排除アークリストrem があるため、O(n2) の記憶スペースが必要である。また、分岐図の葉の最大数は、探索図の深さに対して指数のオーダーとなる。実際の計算では、限定操作により枝刈りが行われるため、やはり上界値関数g と許容解関数l により葉の最大数は変化する。

以上のように、処理時間、処理スペースともに、g,l により変わる。また、意味係りグラフの構造や重み付けの状態により変化する。次章では、実際の文に上記のアルゴリズムを適用し、そのパフォーマンスを検討する。

## 4 解析実験

3.2で述べたアルゴリズムの性質およびパフォーマンスを見るため、論文・マニュアルから100 文を抜き出し解析実験を行った。使用した計算機はエンジニアリングワークステーションAS4260である。また、各文に対する意味係り受けグラフの生成は、既存のソフトウェアを使用した。

表9に測定結果を示す。表の縦方向は、入力文に対する意味係り受けグラフのノード数による分類であり、横方向は、最適解を探索する際に作られた部分問題の個数である。この表の要素には文の出現度数と、それらの文の最適解探索に要した平均時間時間を示してある。また、表中に(x/y) の形式で記述してある部分は、それぞれ次のデータを示している。

x: 最終的に最適解となった許容解を生成した部分問題の番号

y: 最終的に全体として展開された部分問題の数

表9より、この実験において以下の事がいえる。

・分岐された部分問題の数は比較的に少ない

最終的に展開される部分問題数が5以下である文は100 文中93文であり、全体としてかなり低い数値になっている。この理由の一つとして、許容解を求めるアルゴリズムl(Pi)において、グリーディに求めた解が、比較的に良い結果を出し、それによる部分問題の終端が有効に働いたことがあげられる。

・殆どの場合において最適解は計算の初期に得られた

問題解決中に部分問題の分岐を行うが、その最大数は53であった。この文は表中の9aで示した文である。9aでは、展開された部分問題は多かったが、最終的に最適解となった解は、第4番目の部分問題の許容解であった。すなわち、第5番目以降は、結果的に第4番目の部分問題の許容解が最適であることの確認のために展開されたことになる。実際、この実験において、100 文中99文は、5つの部分問題を計算した時点で求まっている。ただ1つ例外であったのは、図中の9bで示した文である。このような結果となった理由としては、意味係り受けグラフの重み付けの結果が非交差条件や多重格禁止条件にあまり矛盾が生じない状態であったということがあげられる。例えば、「彼は本は買う。」という文の意味係り受けグラフにおいて、意味的な理由により「彼」は「買う」の「行為者」にあたる解釈(アークの重み)が最も強く、また、「本」は「買う」の「対象」にあたる解釈が強いという重み付けがされていれば、そのグラフの最大木が最適解と一致する訳である。

今回の実験では、数ms～1300ms程度で、殆どの場合、解を求めることができたが、中には表中の9aのように12秒近く

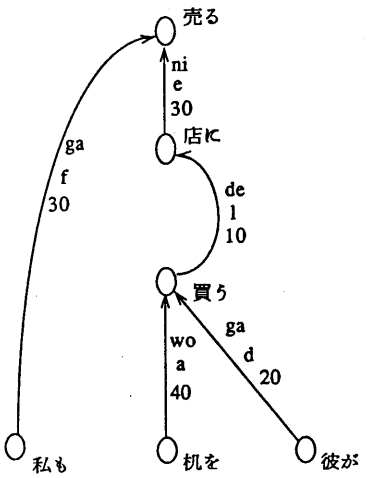



図8 「私も彼が机を買った店に売った。」に対する意味解析木

| 文の頻度 | | | 生成された部分問題数 | | | | |
|---|---|---|---|---|---|---|---|
| | | | 1～5 | 6～10 | 11～15 | 16～25 | 26～ |
| 意味係り受けグラフのノード数 | 1～5 | 9 | 9<br>3ms | | | | |
| | 6～10 | 12 | 12<br>25ms | | | | |
| | 11～15 | 39 | 37<br>80ms | 1(4/7)<br>293ms | 1(4/11)<br>436ms | | |
| | 16～20 | 21 | 20<br>173ms | | 1(1/11)<br>848ms | | |
| | 21～25 | 10 | 7<br>312ms | 1(1/9)<br>1088ms | 1(1/11)<br>1231ms | | 9a |
| | 26～30 | 7 | 6<br>365ms | | | | 1(4/53)<br>11750ms |
| | 31～35 | 1 | | | 1(12/13)<br>3775ms | 9b | |
| | | 1 | 1(1/5)<br>1258ms | | | | |

表9 例文に対する実験結果

(x/y) x:許容解を最初に得た部分問題の番号
y:最終的に展開された部分問題の数

も時間を要する例もあった。9aでは、最適解は、比較的初期(4番目)に求まっていたが、限定操作が旨く適用できず、結局多くの部分問題を生成してしまっている。この様な場合には、解の上限(現状では、最大木の重さ)をもう少し精度の高いものに変えることが有効である。また、乱暴な手段ではあるが、展開する部分問題の数を限定してしまう(もちろん近似解を得ることになる)という方法も有効ではある。しかしこの方法では、図中の9bのような場合に対して問題となる。9aとは対称的に9bでは、部分問題の展開は、最適解を求めるために必要な数だけ行われたのみであった。この例に対して計算の収束を早めるためには、最適解の探索を早めに決定できるようにすることが必要である。

## 5 アルゴリズムの改良

4章で述べた実験では、大部分の例文について数100ms 程度で最適解を得ることができたが、中には、10秒以上も計算時間を要した例もあった。次に、アルゴリズムの改良について検討する。

・上解値関数の改良

既に述べたように、生成される部分問題の数を減らすことは、全体の処理効率を上げる上で重要である。4の結果より、最適解は多くの問題で比較的早い時点で求まっており、その解が最適解であることを確認するための時間がかなり多いことがわかった。これを改善するために、上界値関数g を改善することを考える。3.2のアルゴリズムでは、最大木を上界値としたが、新たなg として、最大木中で共起条件を満たしていないアークのペアにたいしては、安全なペナルティをその全体の重みより差し引くという改良が考えられる。例えば、アークSi[k] とSj[l] が共起条件を満足しない場合に、

min(w(Si[k])-w(Si[k+1]),w(Sj[l])-w(Sj[l+1]))
(wはアークの重み)

を最大木の重みから差し引くといった改良である。ただ、この新たなg は、O(n2) の計算を必要とし、最大木の計算より時間がかかる。このため、全体としてのトレートオフを考える必要がある。

・部分計算の共有化

部分問題をその子供の問題に展開する際に、共起条件を満足しない2つの問題に展開する。親の問題と子供の問題は、かなり似た問題である。子供の問題の計算の際に、親の問題での計算の部分を子供の問題の計算に利用することが考えられる。ここで提案したアルゴリズムは、許容解の計算にバックトラックベースのアルゴリズムを使用しており、効率が良くない場合がある。このため、許容解の計算を親子で共有することが有効であると思われる。

実際に上記の2つの改良を施したアルゴリズムにより、図9の9aの文を解析したところ、生成された部分問題数は、53->9となり、実行時間は、11750ms->1326msと大幅に改善された。また、9bの問題は、生成された部分問題数が、13->11となり、解析時間は、3775ms->2826msへと改善された。

・制約伝搬処理の導入

このアルゴリズムの問題である意味係り受けグラフには、局所的な計算により、可能性を排除できるアークが存在する。[7] で述べられているような、制約伝搬アルゴリズムを用いることにより、グラフの探索空間を縮小させることが可能である。

## 6. おわりに

本報告では、自然言語の各種レベルの知識の性質(優先性を現す場合が多い。異なったレベルの知識の干渉がある)に合致した処理方式へのアプローチとして、構文／意味レベルの曖昧性を処理するワク組みとその実験について報告した。この種の処理では、最終的には組み合わせ的な計算を行う必要があり、その手法として分岐限定法を用いている。今回対象としたのは、構文／意味レベルの知識であるが、同様の議論はその他のレベルにもあてはまる。例えば、形態素解析における解釈の曖昧性が、意味や文脈レベルの知識と関係を持つような場合である。特に、英語の解析においては、多品詞語が非常に多く存在するため、品詞の解釈を他の知識と整合よく行う必要があり、かなり広い解空間に対し制約／優先処理を高速に行う方式が必要となる。

【謝辞】

アルゴリズムの検討に協力していただいた、情報システム研究所の磯部氏、ならびに小野氏に感謝致します。

【参考文献】

[1] 奥村、田中:自然言語解析における意味的曖昧性を増進的に解消する計算モデル,情報処理学会,自然言語処理研究会資料71－1,1989

[2] 杉村:論理型文法における制約解析,情報処理学会,自然言語処理研究会67－2,1988

[3] 平川、天野:構文／意味優先規則による日本語解析,人工知能学会,第3回全国大会論文集,1989

[4] Tomita,M: An efficient augmented context-free parsing algorithm,Computational Linguistics Vol.13, 1987

[5] Barton,G.E.and Berwick,R.C.: Parsing with Assertion Sets and Information Monotonicity, Proceedings of International Joint Conference of Artificial Intelligence-85, 1985

[6] Seo,J and Simmons.R.F:A Syntactic Graphs: A Representation for the Union of All Ambiguous Parse Trees, Computaional Linguistics Vol.15,1989

[7] 丸山:制約依存文法に基づいた日本語解析支援システム,情報処理学会,自然言語処理研究会資料69-6,1988.12

[8] 茨木:組み合せ最適化,産業図書出版,1983