自然言語解析
—整数計画問題としての定式化と解法—

鈴木　潤

文または文章に，文法・意味的な情報を付与する処理を行う自然言語解析技術は．計算機の演算能力の向上や離散最適化技術の成熟を背景に，10 年ほど前（2000 年代半ば以降）から，整数計画問題に基づく定式化とその解法を利用した方法論が盛んに研究されるようになった．本稿では，自然言語解析における整数計画法に基づく定式化を中心に，最適化問題としての自然言語解析の性質と，これまでに考案されてきた解法を紹介する．

キーワード：自然言語処理，自然言語解析，動的計画法，整数計画問題，構造予測，双対分解

1. はじめに

「自然言語解析」とは，「文または文章に，文法的または意味的な情報を付与する処理」を総称する用語として用いられている．自然言語解析（特に構文解析）は，機械翻訳と並んで最も古くから取り組まれている自然言語処理研究の最重要項目である．

自然言語解析は，さらに，形態素分割（分かち書き），品詞付与，固有表現抽出，係り受け解析，談話構造解析，といったさまざまな種類に細分化される．これらの分類は，文法，意味論，語用論といった言語学的な知見に基づいて決められている部分もあるが，主に計算機による自動解析の研究の流れの中で，所望するすべての情報を一括して解決することは困難なので，解析問題を細分化して独立の問題として個別に取り組むために分類したという歴史的経緯に起因している部分が大きいと考えられる．

自然言語解析の研究では，まず第一に，言語学的知見，あるいは，実現したいシステムに必要な要件などをすべて考慮したうえでどのような情報を自然言語の文章に付与すべきかという「問題の定義」をし，次に，その定義に従って「人手により（一定量の）データを作成」する．この二つの工程を終えると，以降は，定義やデータに従った効率的な解析アルゴリズムの研究が行われる．自然言語処理分野の研究という観点では，問題の定義とデータ作成のフェーズが最も重要かつ時間もコストもかかる工程となる．しかし，本稿では，問題が与えられた後のモデル化とアルゴリズム研究に焦点を当て，最初の二つの最重要工程は事前に終えていることを前提に話を進める．

たとえば，同じ条件でよりよいアルゴリズムを設計する研究を行う場合は，いかに時間・空間計算量を削減できるかが議論の中心となる．また，別の観点として，さらなる性能向上のため，これまで容易に扱えなかった情報（特徴）を新たに取り入れるモデル化の議論を行う場合は，追加する情報を計算量的にいかに効率よく扱うことができるかが争点となる．

自然言語解析のアルゴリズム研究は，古典的には，効率的な動的計画法の設計の歴史と言い換えても語弊がないほど動的計画法に基づく方法論が大半を占めている．動的計画法は，局所的な最適解をたどることで大域的な最適解を得る方法のため，問題を分割して効率的に解くことができる反面，その性質として大域的な制約や条件を扱うのが難しいという欠点がある．しかし 10 年ほど前から，計算機の演算能力の向上や最適化技法の成熟により，大域的な情報を扱う最適化や，複数の解析問題を一括して解くような方法論が議論されるようになってきた．本稿では，特にここ 10 年で急速に研究が進んだ，整数計画法に基づく自然言語解析に着目し，定式化と実際に考案されてきた解法を紹介する．

2. 自然言語解析の概要

本節では，自然言語解析の概要を述べる．自然言語解析の種類に関しては，本特集にある「自然言語処理概論」に解説があるため，ここでの説明は割愛する．

2.1 利用可能な公開実験データ

国際会議 CoNLL (The SIGNLL Conference on Computational Natural Language Learning[1]) にお

すずき　じゅん
日本電信電話株式会社 コミュニケーション科学基礎研究所
〒 619–0237 京都府相楽郡精華町光台 2–4
suzuki.jun@lab.ntt.co.jp

[1] http://www.conll.org

表 1　CoNLL shared task の年ごとのまとめ

年	内容（タスク）
1999–2001	名詞句・フレーズ分割タスク（英語）
2002, 2003	固有表現抽出（英語，ドイツ語， スペイン語，オランダ語）
2004, 2005	意味役割付与（英語）
2006, 2007	係り受け解析 （英語，日本語含む 7 から 13 カ国語）
2008, 2009	係り受け解析＋意味役割付与 （英語，日本語含む約 13 カ国語）
2010	領域検出（英語）
2011, 2012	共参照解析（英語，中国語，アラビア語）
2013, 2014	文法誤り訂正（英語）
2015, 2016	談話構造解析（英語，中国語）

いて，2000 年頃から shared task という自然言語解析システムのコンペティション（同一条件下でのシステムの順位付け）が毎年開催されている．ここで毎年整備されリリースされるデータが，自然言語解析研究のデファクトのベンチマークデータとして多くの論文の実験に使われている．表 1 に，shared task の年ごとのタスクをまとめる．多くのデータは研究目的であれば無料で利用できる[2]．また，LDC (Linguistic Data Consortium[3]) からライセンスを購入することで，利用可能となるデータも一部存在する．

CoNLL データは，主にテキスト形式で配布されており，フォーマットが毎年おおむね同じため，利便性が高く，前処理不要で実験に利用できる非常に扱いやすいデータである．

2.2　構造予測問題

前述の CoNLL データは，会議の名前からもわかるように，機械学習を用いた自然言語解析の研究に用いることが主な目的である．つまり，離散最適化のアルゴリム研究を目的としてデータが作成されているわけではない．しかし，自然言語解析の問題そのものは，離散構造を予測する問題とみなすことができる．機械学習の分野では，このような問題を「構造予測 (Structured Prediction) 問題」と呼び，比較的難しい問題に分類している．また，構造予測問題は複雑な制約付きの最適化問題となるため，それぞれの問題に適した効率的な解析アルゴリズムを開発する必要がある．

$\boldsymbol{x}$ を入力された文または文章，$\mathcal{Y}(\boldsymbol{x})$ を入力 $\boldsymbol{x}$ から生成可能な出力の候補集合（制約を満たした候補集合）とする．まず，係り受け木（詳細は 3 節で説明する）$\boldsymbol{y}$ に対する「部分構造」を定義する．部分構造の定義は，用いる解析アルゴリズムに附随して決まるため一概には言えないが，係り受け解析においてもっとも簡単な部分構造の例は，2 単語間の一つの係り受け関係である．次に，可能なすべての部分構造を列挙し，一意に決まる番号を割り振る．すると，係り受け木 $\boldsymbol{y}$ は，その中に含まれる部分構造の番号の集合 $\delta(\boldsymbol{y})$ で表現することができる．このとき，構造予測問題で用いる最適化問題は，以下の汎用的な形式で表すことができる．

$$\hat{\boldsymbol{y}} = \mathop{\arg\max}_{\boldsymbol{y}\in\mathcal{Y}(\boldsymbol{x})} \left\{ \sum_{r\in\delta(\boldsymbol{y})} s_r \right\} \tag{1}$$

ただし，s_r を r 番目の部分構造に対するスコア，$\hat{\boldsymbol{y}}$ を予測結果の構造（出力）とする．

つまり，この最適化問題は，部分構造のスコアの総和が最も大きい出力 $\hat{\boldsymbol{y}}$ を出力候補集合 $\mathcal{Y}(\boldsymbol{x})$ から選択する問題である．実際には，$\mathcal{Y}(\boldsymbol{x})$ や部分構造としてどのような情報を利用するかを事前に決める必要がある．また，その選択の仕方によって，用いられる解析アルゴリズムが変わってくる．

次に，解の候補集合 $\mathcal{Y}(\boldsymbol{x})$ 中に含まれる解候補の部分構造の種類数が R 個だったとする．また，$\boldsymbol{z}$ を R 次元の二値ベクトルとする．このとき，式 (1) の最適化問題を整数計画問題として再定式化すると，以下のようになる．

$$\begin{aligned} \hat{\boldsymbol{z}} = \mathop{\arg\max}_{\boldsymbol{z}} & \left\{ \sum_{r=1}^{R} s_r z_r \right\} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{z} \le \boldsymbol{b} \\ & \boldsymbol{z} = (z_1, \ldots, z_R)^\top \in \mathbb{B}^R \end{aligned} \tag{2}$$

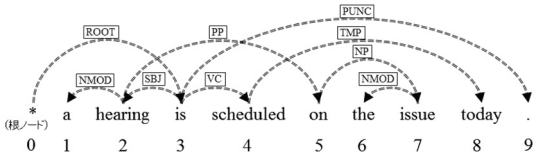
ただし，$\boldsymbol{A}$ は $\boldsymbol{z}$ 間の制約関係を表すために用いる係数の行列，$\boldsymbol{b}$ は各制約の補正項のベクトル表記，$\mathbb{B}$ は 0 または 1 の二値を表す ($\mathbb{B} = \{0, 1\}$)．つまり，もし制約が C 個の場合，$\boldsymbol{A} \in \mathbb{R}^{C\times R}$, $\boldsymbol{b} \in \mathbb{R}^C$ となる．

式 (1) では，解候補集合 $\mathcal{Y}(\boldsymbol{x})$ に含まれる解 $\boldsymbol{y}$ が何かしらの方法で得られるという前提で定式化が行われている．一方，式 (2) では，解の満たすべき制約，いわゆる解空間を $\boldsymbol{A}\boldsymbol{z} \le \boldsymbol{b}$ という形で一括して記述する方法で定式化している．よって，得られた $\hat{\boldsymbol{z}}$ は，値が 1 となる要素番号に対応する部分構造をもってくることで，もとの構造 $\hat{\boldsymbol{y}}$ へ一意に変換することができる．

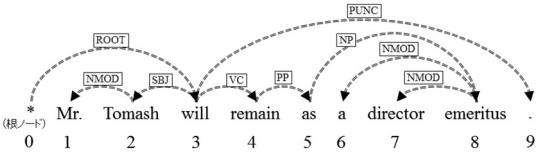
3.　係り受け解析

本稿では，整数計画問題に基づく定式化とその解析アルゴリズムの研究が最も盛んに行われた係り受け解析問題を取り上げ，詳細な説明を行う．

[2] http://www.conll.org/previous-tasks
[3] https://www.ldc.upenn.edu



(a) 交差あり係り受け



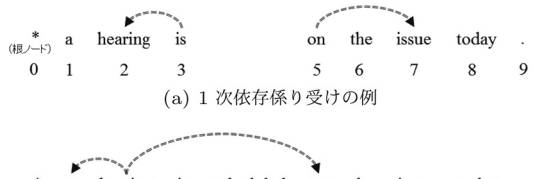
(b) 交差なし係り受け

図 1　係り受け解析の（人手により付与された）正解解析結果の例：文献 [1] からの引用

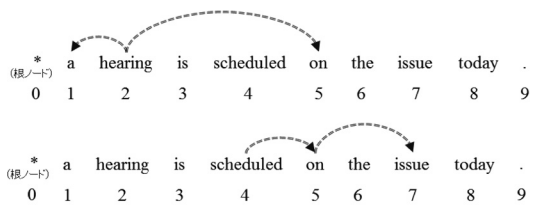
3.1　用語と簡単な概要

係り受け解析とは，与えられた文に出現する単語間の修飾・被修飾関係を推定する処理である．図 1 に係り受け解析の（人手により付与された）正解解析結果を例示する．係り受け解析が，離散最適化問題の観点で面白いと思う点は，文中の単語を頂点（ノード），各単語間の関係を有向辺とみなし，与えられた文中の全単語の関係が必ず木構造となるように解析することが条件となる点である．よって，係り受け解析とは，「木構造条件を満たしつつ，最も適切に単語間の修飾・被修飾関係を決める処理」と言える．また，解析結果が木構造であることから，解析結果のことを一般に「係り受け木（あるいは，係り受け解析木）」と呼ぶ．

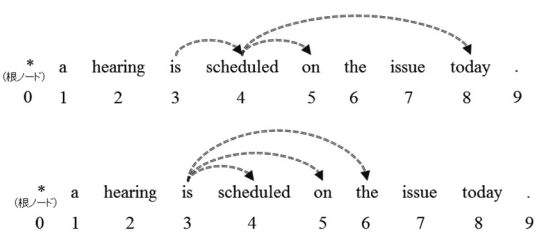
係り受け解析は，扱う問題や解き方などによっていくつか細分類があり，専用の用語が使われている．まず，図 1 に示したように，言語的な特徴に応じて係り受けが交差しないことを前提とする「交差なし (projective) 係り受け」と，交差することを許容する「交差あり (non-projective) 係り受け」の二種類が定義されている．また，図 1 中の「ROOT」や「SBJ」といった係り受け関係のラベルも合わせて予測する「ラベルあり係り受け」と，係り受け関係のラベルは忘れて係り受け関係のみを予測する「ラベルなし係り受け」がある．さらに，解き方として，2 単語間の局所的な関係の情報（つまり一つの係り受け関係）のみを考慮する場合は「1 次依存係り受け」，3 単語間の依存関係（つまり係り受け関係二つ）までの情報を考慮する場合は「2 次依存係り受け」といったように，$k+1$ 個の単語関の情報を用いて解析する場合は「k 次依存係り受け」と呼ぶ．図 2 に 1 次，2 次，3 次の係り受け関係の例を示す．



(a) 1 次依存係り受けの例



(b) 2 次依存係り受けの例：
上：兄弟の依存関係，下：孫の依存関係



(c) 3 次依存係り受けの例：
上：孫兄弟の依存関係，下：3 兄弟の依存関係

図 2　係り受け解析の次数により扱う関係の違いを表した例

係り受け解析では，周辺の部分構造の予測結果に依存して正解が変わる相互依存性と呼ばれる性質がある．そのため，単純に 1 次依存係り受け（二単語間のスコア）のみを考慮して処理をするのでは不十分で，2 次，3 次とより高次の関係を利用したい．一方，高次の情報を利用する場合，計算コストが急激に高くなるというトレードオフの関係にあるため，無制限には次数を上げられない．そのことから，計算コストを極力抑えて次数を上げていくアルゴリズム研究が長年行われ，アルゴリズムがどのレベルの情報を利用しているかを明示するために，係り受け関係の次数を明記するのが慣習となっている．

3.2　動的計画法に基づく定式化

交差あり／交差なしは，問題のクラスとしては，交差ありが交差なしを包含するので，交差なしのほうが，より制約が強い解析をすることになる．解析アルゴリズムの観点では，交差あり／交差なしは，それぞれ計算量的な困難さが違うため，明確に別の問題として扱う場合が多い．たとえば，1 次依存係り受け解析では，交差ありの場合は最大全域木 (maximum spanning tree) を求める問題に帰着できるため，Chu-Liu-Edmonds アルゴリズム [2, 3] を用いることで，単語数 N に対して計算量は $\mathrm{O}(N^2)$ となるが，交差なしの場合は，交差しないことを制御するために最低でも Eisner アルゴリズム [4] の $\mathrm{O}(N^3)$ 以上の計算量を必要とする [5].

表 2　ラベルなし係り受け解析の次数に対する動的計画法に基づく解析アルゴリズムに必要な計算量（ラベルあり解析の場合は，それぞれラベル数 L だけ計算量が増加する）

	交差なし			交差あり		
	時間	空間	文献	時間	空間	文献
1 次	$O(N^3)$	$O(N^3)$	[4]	$O(N^2)$	$O(N^2)$	[5]
2 次	$O(N^4)$	$O(N^3)$	[6]	（近似）	$O(N^3)$	[7]
3 次	$O(N^4)$	$O(N^3)$	[8]		N/A	
4 次	$O(N^5)$	$O(N^4)$	[9]		N/A	
それ以上	（近似）	$O(N^3)$	[10]		N/A	

一方，2 次依存係り受け解析では，交差なしの場合は $O(N^4)$ の計算量でアルゴリズムを構築することができるが [6]，交差ありの場合は，NP 困難の問題であり，多項式時間で解けるアルゴリズムは現時点で知られていない [7].

表 2 に，ラベルなし係り受け解析の次数に対する動的計画法に基づく解析アルゴリズムに必要な計算量を簡単にまとめる． 本稿が着目する整数計画法に基づく解析アルゴリズムは，主に，動的計画法ではうまく扱うことができない，2 次以上の交差あり係り受け解析において大きく発展した.

3.3　整数計画法による定式化

自然言語解析問題は，相互依存性という性質があるので，なるべく高い次数の依存関係を利用して解析を行いたい．実際，次数が上がるたびに解析精度が飛躍的に向上する傾向が見られる．交差なしという条件があれば，動的計画法に基づく解析アルゴリズムで，比較的高次の依存関係を考慮した解析が多項式時間で実現できることは前に述べたとおりである．一方，交差ありの場合は，動的計画法の枠組みでは，高次の依存関係を効率的に取り扱うのは困難であるため，別の方法論として，整数計画法に基づく解析アルゴリズムが取り上げられるようになった.

まず，高次の依存関係に対する定式化を述べる前に，整数計画法の枠組みで木構造であることをどう効率的に判定するかについて述べる．定式化のため，まず，$\mathcal{V}$ を頂点番号の集合とする．頂点番号は，0 を根ノードを表す特別な番号とする．仮に単語数が 4 の文の場合，$\mathcal{V} = \{0, 1, 2, 3, 4\}$ となる．次に，頂点集合 $\mathcal{V}$ に対する完全グラフ $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ を考える．ただし，辺集合 $\mathcal{A}$ は有向辺とし，$(i, j) \in \mathcal{A}$ の場合，頂点番号 i から j への有向辺とする．このとき，$\mathcal{A}$ に対する部分集合 $\mathcal{A}' \subseteq \mathcal{A}$ により構成される部分グラフ $\mathcal{G}' = (\mathcal{V}, \mathcal{A}')$ が木構造になっているかどうかを判定したい．理解を助けるため，図 1(a) に示した係り受け関係を行列表記したものを図 3 に示す．ここでは，係り受け関係にある要素を 1，係り受け関係にない要素を 0 で表す．また，それぞれの要素は 0 または 1 以外の値は取らないこととする．ただし，可読性を上げるため，図 3 には「係り受け関係なし」を表す 0 は表記していない（空欄になっている要素の値はすべて 0 である）． ここで，各列は各頂点の入力辺の情報，各行は各頂点の出力辺の情報を表している.

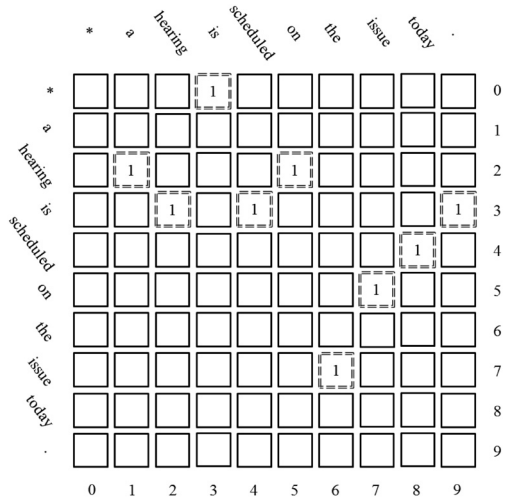


図 3　図 1(a) に示した係り受け関係（係り受けラベルなしの場合）を行列表記したもの（係り受け関係のある：1，ない：無表記）

係り受け解析の結果が木構造となるために満たすべき条件は以下の三つで記述できる.

1. 根ノード $0 \in \mathcal{V}$ は入力辺を一つももたない
2. 根ノード以外のすべての頂点 $\forall v \in \mathcal{V}\backslash\{0\}$ は入力辺を必ず一つもつ（一つの親となる頂点をもつ）
3. サイクルをもたない

1 番目と 2 番目の条件は比較的簡単に判定することができる．条件 1 に関して，図 3 の行列表記を用いて定式化する場合，一列目の合計が 0 であることで表現できる.

$$\sum_i z_{i,0} = 0 \tag{3}$$

ただし，$z_{i,j}$ は行列の (i, j) 成分の値を表し，$z_{i,j} \in \mathbb{B}$ である．また，条件 2 に関しては，図 3 の一列目以外の列方向の合計が必ず 1 となるようにすればよい.

$$\sum_i z_{i,v} = 1 \quad \forall v \in \mathcal{V}\backslash\{0\} \tag{4}$$

ただ，3 番目のサイクル条件の判定は比較的困難である．仮に愚直に判定を行う場合，空間計算量が指数と

なり，効率的な計算が難しい．そのため，係り受け解析に最も早く整数計画法を適用した論文 [11] では，切除平面法の考えに基づいて，最初は木構造制約なしの状態から最適解の探索を行い，サイクルを含む解（つまり木構造条件を満たさない解）が見つかった場合，その解を除去する制約を逐次的に追加して再度探索を行い，最初に木構造制約を満たす解が見つかったら，それが求める最適解である，という方法で係り受け解析を行った．

次に，文献 [12] では，3 番目のサイクル条件を「グラフが連結か」という判定条件へと変更することで，空間計算量を多項式に削減できることを示した．より具体的には，グラフが連結かどうかの判定は，根ノードからほかのすべての頂点に向かってちょうど一つずつの荷物を届けるフロー問題 (single-commodity flow) の制約と等価であることを示した．また，このフロー問題は，空間計算量 $\mathrm{O}(N^2)$ で処理可能であることが広く知られている．つまり，前述のサイクル条件の判定に必要な空間計算量が指数から多項式へと大幅に削減することができた．

以下に，フロー問題の定式化による木構造の判定に用いる制約式を述べる．まず，図 3 の行列と同様な $(N+1)\times(N+1)$ の行列を考える．次にその (i,j) 要素 $h_{i,j}\in\mathbb{N}$ は頂点 (i,j) 間を輸送する物量を表す．ただし，$\mathbb{N}$ を 0 を含む自然数の集合とする．このとき，根ノードは，すべての頂点に荷物を一つずつ届けるため，出発点として総物量は頂点数 N になる．

$$\sum_j h_{0,j}=N \tag{5}$$

次に，根ノード以外のある v 番目の頂点では輸送された荷物を一つ受け取ると考えるので，以下のような「入ってくる物量と出て行く物量の差は必ず 1 になる」という制約で表現することができる．

$$\sum_i h_{i,v}-\sum_j h_{v,j}=1 \quad \forall v\in\mathcal{V}\backslash\{0\} \tag{6}$$

最後に，すべての辺 $h_{i,j}$ において，輸送路となる辺以外は必ず物量が 0 となり，輸送路となる辺は 1 から N までの値をとるという条件を以下のように表す．

$$h_{i,j}\le Nz_{i,j} \quad \forall(i,j)\in\mathcal{A} \tag{7}$$

式 (7) での輸送路とは，つまり木構造となる有向辺のことなので，$z_{i,j}=1$ となる辺である．一方，輸送路以外の辺では $z_{i,j}=0$ となり，不等式の右辺が 0 となるので，左辺の $h_{i,j}$ も必ず 0 となる．

最後に，$z'_{i,j,\ell}\in\mathbb{B}$ を導入し，以下の式を用いて 0 を含む自然数 $h_{i,j}\in\mathbb{N}$ を二値 $z'_{i,j,\ell}\in\mathbb{B}$ に置き換えて表す．

$$\begin{aligned} h_{i,j}&=\sum_{\ell=1}^{N} z'_{i,j,\ell}\\ z'_{i,j,\ell-1}&\ge z'_{i,j,\ell} \quad \forall(i,j,\ell)\in\mathcal{C}\\ \mathcal{C}&=\{(i,j,\ell)\mid 0\le i,j\le N,\ 2\le\ell\le N\} \end{aligned} \tag{8}$$

最終的に，式 (5), (6), (7), (8) を合わせて，1 次依存係り受け解析は以下のような定式化ができる．

$$\begin{aligned} \hat{\boldsymbol{z}}&=\arg\max_{\boldsymbol{z},\boldsymbol{z}'}\left\{\sum_r s_r z_r\right\}\\ \text{s.t.}\quad &\boldsymbol{A}\begin{bmatrix}\boldsymbol{z}\\ \boldsymbol{z}'\end{bmatrix}\le\boldsymbol{b},\quad \boldsymbol{z}\in\mathbb{B}^R,\quad \boldsymbol{z}'\in\mathbb{B}^{RN} \end{aligned} \tag{9}$$

ただしここでは，$z_{i,j}$ といった二次元の要素番号 (i,j) を，たとえば $i+j\times(N+1)+1=r$ を用いて一意に一次元の要素番号 r に変換し，ベクトル $\boldsymbol{z}$ を構成していると仮定する．同様に，$\boldsymbol{z}'$ も，$z'_{i,j,\ell}$ を任意の要素番号の変換を用いて一次元に並べたベクトルとする．

ここで注意すべき点は，補助変数 $z'_{i,j,\ell}$ に対するスコアがすべて 0 であるとみなせば，目的関数に $z'_{i,j,\ell}$ は現れないので，式 (9) は式 (2) の形式で表現されていることである．また，二値の変数のみで定式化を行った場合は，空間計算量は $\mathrm{O}(N^3)$ になる．

3.4 高次の係り受けの導入

文献 [12] では，2 次依存関係を含めた整数計画問題としての定式化を提案している．まず，図 2(b) の兄弟関係にある 2 次依存関係を導入するために，i 番目の頂点を親とし，j 番目と k 番目の頂点を兄弟とする三つの単語間の関係を表す補助変数 $z^{\mathrm{sib}}_{i,j,k}\in\mathbb{B}$ を導入する．このとき，以下の不等式で表される制約を追加して，図 2 で示している兄弟関係にある 2 次依存関係を表す．

$$\begin{aligned} &z^{\mathrm{sib}}_{i,j,k}\le z_{i,j},\quad z^{\mathrm{sib}}_{i,j,k}\le z_{i,k},\\ &z^{\mathrm{sib}}_{i,j,k}\ge z_{i,j}+z_{i,k}-1 \quad \forall(i,j,k)\in\mathcal{C}'\\ &\mathcal{C}'=\{(i,j,k)\mid(i,j)\in\mathcal{A},(i,k)\in\mathcal{A}\} \end{aligned} \tag{10}$$

同様に，孫関係にある 2 次依存関係を導入するには，以下の不等式で表される制約を追加する．

$$\begin{aligned} &z^{\mathrm{gp}}_{i,j,k}\le z_{i,j},\quad z^{\mathrm{gp}}_{i,j,k}\le z_{j,k},\\ &z^{\mathrm{gp}}_{i,j,k}\ge z_{i,j}+z_{j,k}-1 \quad \forall(i,j,k)\in\mathcal{C}''\\ &\mathcal{C}''=\{(i,j,k)\mid(i,j)\in\mathcal{A},(j,k)\in\mathcal{A}\} \end{aligned} \tag{11}$$

ただし，$z^{\mathrm{gp}}_{i,j,k}\in\mathbb{B}$ は，i 番目の頂点を親とし，j 番目の頂点を子，k 番目の頂点を孫とする三つの単語間の

関係を表す補助変数とする．

式 (10) と式 (11) は同じように見えるが，(i,k) 要素を用いる部分と (j,k) 要素を用いる部分に違いがある．たとえば式 (10) の場合で制約の意味を解釈してみる．補助変数 $z^{\mathrm{sib}}_{i,j,k}=0$ のときは，$z_{i,j}$ と $z_{i,k}$ が両方 0 か，いずれかが 1 ならばすべての制約を満たすが，$z_{i,j}$ と $z_{i,k}$ の両方が 1 となる場合は，制約を満たさなくなる．つまり，$z_{i,j}$ と $z_{i,k}$ の両方が 1 の場合は，必ず $z^{\mathrm{sib}}_{i,j,k}=1$ でなくてはならないという関係を表している．

最後に，補助変数 $z^{\mathrm{sib}}_{i,j,k}$ に対するスコア $s^{\mathrm{sib}}_{i,j,k}$ や，$z^{\mathrm{gp}}_{i,j,k}$ に対するスコア $s^{\mathrm{gp}}_{i,j,k}$ を新たに導入することで，2 次依存関係のスコアを加味した係り受け解析が可能となる．式 (10) や式 (11) からも明らかなように，補助変数を導入することで必要となる空間計算量は $\mathrm{O}(N^3)$ である．

3.5 整数計画問題からの次の発展

前節で示した整数計画問題としての定式化の場合，実際の解法には，汎用の整数計画ソルバー（CPLEX, Gurobi など）を利用していた．ただ，汎用の整数計画ソルバーでは，従来法に相当する動的計画法に基づく方法より，明らかに計算速度が遅くなるので，それを改善するために文献 [13, 14] では，双対分解を用いた高速な専用解法を提案している．特に，文献 [14] で述べられている方法は，現在でも新しい手法の性能を評価する際に基準となる安定的な手法の一つとして以降の論文で度々比較対象として利用されている方法論である．

また，文献 [15] では，列生成法 [16, 17] を適用する方法を提案している．これは，文献 [13, 14] と同様に，高速化と省メモリ化を目的に最適化に使う変数を逐次増やしながら繰り返し最適化を行う方法となっている．扱っている高次の依存関係は，ほぼ同じと考えてよいので，単純に計算速度と省メモリ化に関する方法と考えてよい．

また，係り受け解析問題とは違うが，状況はおおむね似た機械翻訳の解析アルゴリズムとして文献 [18] では，主問題に対する双対問題を上限値として利用し，主双対ギャップを計算しながら，反復的にビームサーチを行うことで，最適な解析結果を求める方法を提案している．

このように，離散最適化研究で培われてきた技術を活用して，独自の解析アルゴリズムの構築を行っている．

4. 自然言語解析の難しさ

ここで，離散最適化研究の一つの適用先として，自然言語解析を利用する状況を考える．つまり，新たに考案した最適化アルゴリズムの効果を示すベンチマークデータとして，自然言語解析のデータを利用する．このような状況で考慮すべきいくつかの注意点を述べておく．

4.1 各決定変数 z_r に対応するスコア s_r の学習

これまでの議論では，各決定変数 z_r に対応するスコア（あるいは尤度）s_r は事前に与えられていることを前提として話を進めてきた．しかし，実際には，スコア s_r は事前に与えられない．たとえば，離散最適化アルゴリズム研究でよく扱われる実データの場合，人や物といった物理的な情報が背景にあることで，スコアも人手により与えられることが多い．一方，自然言語解析の場合，そもそもどのような状態や特徴がどのくらいのスコアに相当するかは専門家であっても容易に決められない問題である．

そのため自然言語解析では，大量の特徴抽出関数を用意し，それらの重み付き総和を用いてスコア s_r を計算する．ここで，特徴抽出関数および重みの総数を D とする．また，d 番目の特徴抽出関数を $\phi_d(\cdot)\in\mathbb{B}$，また，$d$ 番目の特徴抽出関数から得られた特徴に対応する重みを $\boldsymbol{w}_d\in\mathbb{R}$ とする．このとき，スコア s_r を以下の式を用いて計算する．

$$s_r=\sum_{d=1}^{D} w_d\phi_d(\boldsymbol{x},r) \tag{12}$$

ただし，$\boldsymbol{x}$ は入力文を表し，r は出力候補の部分構造の要素番号を表すとする．

次に，$\boldsymbol{w}=(w_1,\ldots,w_D)^\top\in\mathbb{R}^D$ を w_d を 1 から D まで並べてベクトル表記したものとする．そして，人手により作成された自然言語解析の正解データ $\mathcal{D}=\{(\boldsymbol{x}_i,\boldsymbol{y}_i)_{i=1}^I\}$ を用いて，教師あり学習を使って $\boldsymbol{w}$ を決定する．よく用いられる定式化は，以下の連続最適化問題を用いてパラメタを推定する．

$$\hat{\boldsymbol{w}}=\arg\min_{\boldsymbol{w}}\{\Psi(\boldsymbol{w},\mathcal{D})+\Omega(\boldsymbol{w})\} \tag{13}$$

ただし，$\Psi(\cdot)$ は，正解データと現在のパラメタ $\boldsymbol{w}$ を使って推定した解析結果がどのぐらい間違えているかを計算するリスク関数である．また，$\Omega(\boldsymbol{w})$ は，過学習を防ぐために導入される正則化項である．

リスク関数や正則化項はそれぞれ状況に応じて選択されるものである．例として，以下のような式を用いる．

$$\begin{aligned}\Psi(\boldsymbol{w},\mathcal{D})&=\sum_{(\boldsymbol{x}^*,\boldsymbol{y}^*)\in\mathcal{D}}\max\Big(0,E(\boldsymbol{y}^*,\hat{\boldsymbol{y}})\\&\qquad+S(\hat{\boldsymbol{y}})-S(\boldsymbol{y}^*)\Big)\\S(\boldsymbol{y})&=\sum_{r\in\delta(\boldsymbol{y})}s_r\end{aligned} \tag{14}$$

ただし，$E(\boldsymbol{y}^*,\hat{\boldsymbol{y}})$ は，正解 $\boldsymbol{y}^*$ と現在の推定結果 $\hat{\boldsymbol{y}}$ との差分を表す関数とする．たとえば，係り受け関係を間違えた個数などを用いる．つまり，正解 $\boldsymbol{y}^*$ と現在の

推定結果 $\hat{\boldsymbol{y}}$ が完全に一致すれば，$E(\boldsymbol{y}^*, \hat{\boldsymbol{y}}) = 0$，係り受け関係を二つ間違えていれば $E(\boldsymbol{y}^*, \hat{\boldsymbol{y}}) = 2$ である．

式の詳細はともかく，ここで最も伝えたいことは，離散最適化により決定する $\hat{\boldsymbol{y}}$ が含まれるところである．このことからわかるように，機械学習により離散最適化で用いるスコアの推定をするために，その連続最適化の中で離散最適化問題を何度も繰り返し解かなくてはいけない，ということになる．つまり，自然言語解析のデータを用いて離散最適化アルゴリズムの研究を行うためには，同時に機械学習法を構築することも求められてくる．さらに，出力が単純な回帰や分類問題ではなく，これまで議論してきたような，複雑な制約条件を満たした出力を予測する問題であるため，構造学習問題と呼ばれる教師あり学習の中でも比較的難しい最適化問題を解くことが求められる．

4.2 要求実行速度

情報処理のアルゴリズム研究の常として，計算速度と性能はおおむねトレードオフの関係になる．よって，解析精度を数ポイント上げるために処理時間が 10 倍になるという方法論は，基本的には受け入れられない．最初に述べたように，自然言語解析のアルゴリズム研究は，歴史的には動的計画法の構築に関する研究と言える．そのため，従来の動的計画法の実行速度をそれほど損なわずに，解析精度だけを向上させることが求められる．

実際，近似解法や問題に特化した独自アルゴリズムの研究が近年の主流の方法となっている．具体的に，係り受け解析では，秒間 100 文以上の処理速度を求められるため，凝った処理はほとんど利用することができないという状況である．

このような状況のため，自然言語解析のベンチマークデータでトップスコアを出すのは，かなり難しい状況になりつつある．逆に，このような厳しい競争の中でトップスコアを出すと，インパクトの高い論文という評価を受ける可能性が高いとも言える．

4.3 深層学習／ニューラルネットの台頭

本稿の参考文献を見てもらうとわかるように，自然言語解析の研究で離散最適化技術の活用が盛んに行われていたのは，2010 年から 2014 年頃である．2014 年以降は，自然言語解析の研究でも深層学習／ニューラルネットの導入が急速に進み，解析アルゴリズムは，いったん一昔前のものに戻ってしまったという状況になっている．

実際に，現在の係り受け解析のトップスコアは動的計画法に基づく最もシンプルな最大全域木を求めるアルゴリズムを使っている [19, 20]．このように解析アルゴリズムがいったん衰退した背景には，深層学習のような複雑な特徴抽出関数を適用すると，複雑な解析アルゴリズム側がそれを取り込むことができない，あるいは，非常に実装が困難ということがある．また，シンプルな解析アルゴリズムでも，トップスコアを出せるということが実証されたことにも起因すると考えられる．しかし一方で，技術が進歩することで複雑な解析アルゴリズムにも適用可能な深層学習の方法論が開発されるのは確実と考えられる．よって，おそらく数年後に深層学習をベースとした離散最適化アルゴリズムのリバイバルがやってくるのではないかと予想できる．

5. おわりに

本稿では，自然言語解析の中で特に係り受け解析を中心に，整数計画問題による定式化と解析アルゴリズムについて紹介を行った．自然言語解析は，長年の研究成果として比較的豊富に人手により作成された正解データが存在する．これらを離散最適化のアルゴリズム研究のベンチマークデータとして，最適化の研究でもうまく活用できたらと考えている．本稿が，自然言語解析特有の設定や状況を理解し，より実践的な方法論を構築する一助になれば幸いである．

参考文献

[1] R. McDonald and G. Satta, “On the complexity of non-projective data-driven dependency parsing,” In *Proceedings of the Tenth International Conference on Parsing Technologies*, pp. 121–132, 2007.

[2] Y. J. Chu and T. H. Liu, “On the shortest arborescence of a directed graph,” *Science Sinica*, **14**, pp. 1396–1400, 1965.

[3] J. Edmonds, “Optimum branchings,” *Journal of Research of the National Bureau of Standards*, **71B**, pp. 233–240, 1967.

[4] J. Eisner, “Three new probabilistic models for dependency parsing: An exploration,” In *Proceedings of the 16th International Conference on Computational Linguistics*, pp. 340–345, 1996.

[5] R. McDonald, F. Pereira, K. Ribarov and J. Hajic, “Non-projective dependency parsing using spanning tree algorithms,” In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pp. 523–530, 2005.

[6] X. Carreras, “Experiments with a higher-order projective dependency parser,” In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pp. 957–961, 2007.

[7] R. T. McDonald and F. C. N. Pereira, “Online learning of approximate dependency parsing algorithms,” In *Proceedings of the 11st Conference of the European Chapter of the Association for Computational*

Linguistics, pp. 81–88, 2006.
[8] T. Koo and M. Collins, "Efficient third-order dependency parsers," In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1–11, 2010.
[9] X. Ma and H. Zhao, "Fourth-order dependency parsing," In *Proceedings of the 24th International Conference on Computational Linguistics: Posters*, pp. 785–796, 2012.
[10] H. Zhang and R. McDonald, "Generalized higher-order dependency parsing with cube pruning," In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 320–331, 2012.
[11] S. Riedel, R. Çakıcı and I. Meza-Ruiz, "Multilingual dependency parsing with incremental integer linear programming," In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pp. 226–230, 2006.
[12] A. Martins, N. Smith and E. Xing, "Concise integer linear programming formulations for dependency parsing," In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pp. 342–350, 2009.
[13] A. Martins, N. Smith, M. Figueiredo and P. Aguiar, "Dual decomposition with many overlapping components," In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pp. 238–249, 2011.
[14] A. Martins, M. Almeida and N. A. Smith, "Turning on the turbo: Fast third-order non-projective turbo parsers," In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 617–622, 2013.
[15] S. Riedel, D. Smith and A. McCallum, "Parse, price and cut-delayed column and row generation for graph based parsers," In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 732–743, 2012.
[16] R. E. Gomory and P. C. Gilmore, "A linear programming approach to the cutting-stock problem," *Operations Research*, **9**, pp. 849–859, 1961.
[17] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, **53**(6), pp. 1007–1023, 2005.
[18] A. Rush, Y.-W. Chang and M. Collins, "Optimal beam search for machine translation," In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 210–221, 2013.
[19] A. Kuncoro, M. Ballesteros, L. Kong, C. Dyer and N. A. Smith, "Distilling an ensemble of greedy dependency parsers into one mst parser," In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1744–1753, 2016.
[20] T. Dozat and C. D. Manning, "Deep biaffine attention for neural dependency parsing," arXiv: abs/1611.01734, 2016.

