

動的変化する文章を対象とした自然言語解析に適した解析アルゴリズムの考案

鈴木 潤

永田 昌明

NTT コミュニケーション科学基礎研究所

{suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

1 はじめに

形態素解析, 固有表現抽出, 係り受け解析といった自然言語の解析 (本稿では以降「自然言語解析」と総称する) の研究では, 通常一文を一入力とし, 解析を始める前に入力文章全体が既知な状態を暗黙の仮定としてきた. それに対し本稿では, 例えばリアルタイム文章校正システムにおいて, 文章が編集される度に (仮に文章が不完全な状態であっても) その時点での文章の修正候補を逐次提示するような状況を想定し, 解析を始める段階では解析対象となる文章全体は未知であり, 任意の可変長文字列が挿入または削除され文章の状態が更新される度に, その時点での文章を解析し結果を出力する問題設定に取り組む.

こういった問題の定義は様々に可能であるが, 本稿では, 解析対象となる文章が編集終了状態か編集途中状態かに関わらず, その時点の文章に対して最適と思われる解析結果を提示する問題とする. そこで, 本稿での基本的な考えとして, 自然言語解析問題を, 一つの整数計画問題と捉える. 次に, 解析対象となる文章が変化する問題設定を, 離散時刻によって最適化変数や制約が徐々に変化していく一連の整数計画問題の問題系列として定式化する. この一連の整数計画問題の問題系列に対して, 各時刻での個々の整数計画問題の最適解を求めつつ, 問題系列全体を効率よく解くアルゴリズムを構築することが本稿の目的となる.

本稿では, 整数計画問題を解くベースのアルゴリズムとして, 文献 [1, 2] で提案された, DD-ADMM と呼ばれる双対分解に基づく最適化アルゴリズムを採用する. この解法は, 最適化変数が最適化中に増減する環境でも効率よく最適化が可能であることが実験的に知られている [3, 4, 5]. 本稿では更に, 問題系列全体を効率よく解く仕組みの一つとして, 最適化変数の更新タイミングを制御する方法を提案する.

以下, 第 2 節にて, 本稿が想定する自然言語解析の問題設定を簡単に述べる. 次に, 第 3 節で, 対象とする自然言語解析問題を, 時刻で最適化変数や制約が動的に変化する整数計画問題の問題系列で定式化し, 第 4 節で, この問題系列全体に対する効率的なアルゴリズムについて述べる. 最後に文献 [6] のように, Penn Treebank データに対する英語のトークン分割, 品詞付与, チャンキング, (部分) 係り受け解析, 固有表現抽出, 文分割の 6 種の同時解析問題を用いて, 提案法の有効性を検証する.

2 動的変化する文章に対する自然言語解析

まず, 離散的な時刻変数 $t \in \{1, \dots, T\}$ を導入する. 自然言語解析をしたい任意の文章 x は, (離散) 時刻 t 毎に徐々に変化すると仮定する. x_t を時刻 t での解析対象となる文章とする. また x_0 を文章の初期状態とする. この初期状態は, 全く入力されていない空の文章でもよいし, 既に文章が与えられている状態でもよい.

もし, x_{t-1} が「朝が来た」, x_{t-1} が「春は眠い」のように, x_{t-1} と x_t に共通部分がなくなければ, 独立に解析すればよく, 特に議論する必要はないと言える. 本稿では, 例えば, x_{t-1} が「文章を解析する」で, 時刻 t で「問題を」が

| | |
|-----------------------|--|
| r : | 最適化変数の添字 ID, $r \in \mathcal{R}$ |
| z_r : | 添字 ID r の最適化変数, $z_r \in \{0, 1\} \quad \forall r \in \mathcal{R}$ |
| s_r : | 添字 ID r の最適化変数に一つに対応するスコア関数の値 |
| \mathcal{S} : | 全ての $r \in \mathcal{R}$ に対する最適化変数 z_r に一つに対応するスコア関数の値 s_r の集合, $\mathcal{S} = \{s_r\}_{r \in \mathcal{R}}$ |
| c : | 制約の添字 ID, $c \in \mathcal{C}$ |
| \mathcal{R}_c : | 添字 ID c の制約で係数が非零となる最適化変数の添字 ID の集合, $\cup_{c \in \mathcal{C}} \mathcal{R}_c \subseteq \mathcal{R}$ |
| $a_{c,r}$: | 添字 ID c の制約での, 添字 ID r の最適化変数 z_r に対する係数 |
| b_c : | 添字 ID c の制約で使われる定数項 |
| \mathcal{A} : | 制約で用いられる全ての係数の集合, $\mathcal{A} = \{a_{c,r}\}_{r \in \mathcal{R}_c, c \in \mathcal{C}}$ |
| \mathcal{B} : | 制約で用いられる全ての定数項の集合, $\mathcal{B} = \{b_c\}_{c \in \mathcal{C}}$ |
| $\mathcal{Z}^{(t)}$: | 整数計画問題 P_t で扱われる全ての最適化変数の集合, $\mathcal{Z}^{(t)} = \{z_r\}_{r \in \mathcal{R}^{(t)}}$ |

図 1 本稿での変数の一覧と定義

末尾に挿入され, x_t が「文章を解析する問題を」となる状況や, 時刻 t で 2 文字目の「章」を削除して, x_t が「文を解析する」と変化する場合には, x_{t-1} と x_t の文章の大部分は同じ文章で, 文字や単語といった任意の長さの文字が挿入または削除されるような状況を想定する.

このような, 時刻変数 t に依存して解析対象となる文章が変化するときに, 時刻 $t = 1$ から最終状態の時刻 $t = T$ までの全ての文章 x_t に対して, 各時刻で最適と思われる解析結果 y_t を効率よく得るアルゴリズムの構築が本稿の目的となる.

3 動的変化する整数計画問題の問題系列

ここでは, 前節で述べた自然言語解析の問題設定を数式で定式化する. 本稿での基本方針として, 一つの自然言語解析問題を, 一つの整数計画問題で定式化する. 次に, 解析対象となる文章が動的に変化する問題設定を, 離散時刻 t によって最適化変数や制約が徐々に変化する一連の整数計画問題の問題系列とみなす. また, 入力文章 x_{t-1} と x_t の類似性から, 連続する時刻 $t-1$ と t で与えられる整数計画問題 P_{t-1} と P_t 間には, 変数と制約の大部分が共有されていることが想定される. 本稿では, このような性質を持つ整数計画問題の問題系列を総称して「動的変化する整数計画問題系列」と呼ぶ.

\mathcal{R} と \mathcal{C} を, 時刻 1 から T までの整数計画問題 P_t に出現した最適化変数と (線形) 制約の添字 ID の集合とする. また, \mathcal{R} および \mathcal{C} を, \mathcal{R} と \mathcal{C} の要素数とする. 次に, $\mathcal{R}^{(t)}$ および $\mathcal{C}^{(t)}$ を, 時刻 t での整数計画問題 P_t で扱われる最適化変数と制約の添字 ID の集合とする. このとき, $\mathcal{R}^{(t)} \subseteq \mathcal{R}, \mathcal{C}^{(t)} \subseteq \mathcal{C}, \mathcal{R} = \cup_{t=1}^T \mathcal{R}^{(t)}$ と $\mathcal{C} = \cup_{t=1}^T \mathcal{C}^{(t)}$ が成り立つ. 図 1 に, 本稿での変数の一覧と定義を示す. このとき, 本稿で定義する「動的変化する整数計画問題系列」における時刻 t での整数計画問題 P_t は以下のようになる:

$$\begin{aligned}
 (P_t) : & \text{maximize} \quad \sum_{r \in \mathcal{R}^{(t)}} s_r z_r \\
 & \text{w.r.t.} \quad z_r \in \{0, 1\} \quad \forall z_r \in \mathcal{Z}^{(t)} \\
 & \text{s.t.} \quad \sum_{r \in \mathcal{R}_c} a_{c,r} z_r \leq b_c \quad \forall c \in \mathcal{C}^{(t)}
 \end{aligned} \tag{1}$$

用いる制約の詳細に関しては、対象とする問題や、解き方の設計に依存して決定されるものなので、ここでは詳細は省略する。

注意点として、本稿では「動的変数整数計画問題系列」の定義を容易にするために、整数計画問題の一般的な表記とは多少異なる形式を用いて定義を行ったが、特殊な整数計画問題を扱っているわけではない。特に、 $T = 1$ の設定を仮定すると、通常の一つの整数計画問題の式と一致する。

ここでのポイントは、時刻 t に依存せず全ての整数計画問題 P_t で A, B, S が共有されているところである。この定義の仕方により、添字 ID_r や c が同じ場合は、時刻 t に依存せず同じ現れ方をするため、ある時刻 $t-1$ と t で、 $\mathcal{R}^{(t-1)}$ と $\mathcal{R}^{(t)}$ 、 $\mathcal{C}^{(t-1)}$ と $\mathcal{C}^{(t)}$ の要素の重複が多ければ、整数計画問題 P_{t-1} と P_t は類似した整数計画問題となるという性質を持つ事になる。

4 動的変数整数計画問題系列の効率的な解法

整数計画問題は、一般的に直接解くのに計算量的な困難性が伴うため、全ての最適化変数 z_r の値域を、0 または 1 の二値の整数値 ($z_r \in \{0, 1\}, \forall r \in \mathcal{R}$) から、0 から 1 の間の実数値 ($z_r \in [0, 1], \forall r \in \mathcal{R}$) に緩和して問題を解く場合が多い (LP 緩和)。 $T = 1$ での式 (1) の整数計画問題とその LP 緩和問題に対しては、現在では、例えば、CPLEX といった強力な商用のソルバーや、フリーの LP ソルバーなど多数存在しており、解法を知らなくても、それらのソルバーを使ってブラックボックス的に問題を解くことも可能である。

よって、本稿では $T = 1$ の時については特に言及せず、 $T \geq 2$ の時に P_{t-1} と P_t の類似性を用いて効率よく解く方法を考案したい。本稿では、問題系列全体を効率的に解くために、双対分解をベースとした解法を提案する。まず、双対分解による整数計画問題の式変形について述べる。次に、ある時刻 t の整数計画問題 P_t に対する双対分解による解法を述べ、最後に問題系列全体への拡張を述べる。

4.1 双対分解法に基づく整数計画問題の定式化

双対分解による解法の大きな考えは、元の最適化問題を、元の最適化変数の部分集合で構成される部分問題の集合に分解して問題を解くということになる。例えば、 G 個の部分問題に分解するとする。このとき、 G を G 個の整数集合とし、 $g \in G$ のとき、 g を各部分問題の添字 ID として用いる。双対分解のポイントは、この部分問題の集合に分解する際に、元の最適化変数 z_r を複数の部分問題に割り当てるように部分問題を設計するところである。ただし、最適化変数全てが複数の部分問題に割り当てられる必要はないが、各最適化変数は必ず一つ以上の部分問題に割り当てられる必要がある。つまり、 \mathcal{R}_g を部分問題 g に含まれる最適化変数 z_r の添字 ID の集合とすると、ある g と g' に対して、 $\mathcal{R}_g \cap \mathcal{R}_{g'} \neq \emptyset$ 、かつ、 $\bigcup_{g \in G} \mathcal{R}_g = \mathcal{R}$ の二つの条件が満たされる必要がある。

同様に、別の満たすべき条件として、部分問題の集合全体で元の最適化問題で扱われている制約を分割して担当する必要がある。そこで、基本的な部分問題の設計として、各部分問題を元の整数計画問題の制約に基づいて構成する。 \mathcal{C}_g を部分問題 g で使われる制約の集合とし、 $\bigcup_{g \in G} \mathcal{C}_g = \mathcal{C}$ を満たすとする。ただし、制約を持っていない部分問題 ($\mathcal{C}_g = \emptyset$) が存在していても良いし、同じ制約が複数の部分問題で使われていてもよい ($\mathcal{C}_g \cap \mathcal{C}_{g'} \neq \emptyset$)。

最後に、最適化変数 z_r をある部分問題 g に割り当てる際には、各部分問題専用の最適化変数 $z_{g,r}$ に変換して割り当てる。同時に、各部分問題の最適化変数 $z_{g,r}$ は、元の最適化問題の一つの最適化変数 z_r を共有していると考え、必ず同じ値を取る必要があるために、等式制約 $z_{g,r} = z_r$ を用いて結合する、という手続きを用いて、最適化問題そのものを変形する。

次に、「動的変数整数計画問題系列」に適用するために、 G を時刻 1 から T までに出現する全ての部分問題の添字

0. 変数の初期化処理: $k = 1, \alpha_{r,g}^0 = 0 \quad \forall \alpha_{r,g}^0 \in \alpha^0, z_r^0 = 0 \quad \forall z_r^0 \in \mathcal{Z}^{(t),k}, z_{g,r}^0 = 0 \quad \forall z_{g,r}^0 \in \mathcal{Z}_g^{(t),k}$

1. $\mathcal{Z}_g^{(t),k-1}$ と $\mathcal{Z}^{(t),k-1}$ を固定し、 $\alpha^{k-1} \rightarrow \alpha^k$ へ更新
全ての部分問題 g と $r \in \mathcal{R}_g$ に対して $\alpha_{g,r}^k \in \alpha^k$ に対して以下を計算:

$$\alpha_{g,r}^k = \alpha_{g,r}^{k-1} + \eta(z_{g,r}^{k-1} - z_r^{k-1})$$

2. α^k と $\mathcal{Z}^{(t),k-1}$ を固定し、 $\mathcal{Z}_g^{(t),k-1} \rightarrow \mathcal{Z}_g^{(t),k}$ へ更新
各 g に対して以下の最適化問題を解くことで $\mathcal{Z}_g^{(t),k}$ を得る:

$$\begin{aligned} \mathcal{Z}_g^{(t),k} = \arg \max_{\mathcal{Z}_g^{(t)}} & \sum_{r \in \mathcal{R}_g} (z_{g,r} - (z_r + s_{g,r}/\rho - \alpha_{g,r}))^2 \\ \text{w.r.t. } & z_{g,r} \in [0, 1] \quad \forall z_{g,r} \in \mathcal{Z}_g^{(t)}, \\ \text{s.t. } & \sum_{r \in \mathcal{R}_c} a_{c,r} z_{g,r} \leq b_c \quad \forall c \in \mathcal{C}_g \end{aligned}$$

3. α^k と $\mathcal{Z}_g^{(t),k}$ を固定し、 $\mathcal{Z}^{(t),k-1} \rightarrow \mathcal{Z}^{(t),k}$ を更新
全ての $z_r^k \in \mathcal{Z}^{(t),k}$ に対して以下を計算する:

$$z_r^k = \bar{z}_{g,r}^k - \bar{\alpha}_{g,r}^k$$

ただし、 $\bar{z}_{g,r}^k$ と $\bar{\alpha}_{g,r}^k$ は、それぞれ、添字 ID_r を含む部分問題 g 中の $z_{g,r}^k$ および $\alpha_{g,r}^k$ の平均である。

4. 収束判定: 収束と判定されなかったら $k = k + 1$ して処理 1 へ戻る。収束と判定されたら終了
二つの小さな正の実数 ϵ_1, ϵ_2 を事前に設定し、 $r_{\text{Primal}}^k < \epsilon_1$ と $r_{\text{Dual}}^k < \epsilon_2$ を満たした際に収束したと判定する [3]。ただし、

$$r_{\text{Primal}}^k = \frac{\sum_{g \in G^{(t)}} \sum_{r \in \mathcal{R}_g} (z_{g,r}^k - z_r^k)^2}{\sum_{r \in \mathcal{R}^{(t)}} \delta(r)}$$

$$r_{\text{Dual}}^k = \frac{\sum_{r \in \mathcal{R}^{(t)}} \delta(r) (z_r^k - z_r^{k-1})^2}{\sum_{r \in \mathcal{R}^{(t)}} \delta(r)}$$

また、 $\delta(r)$ を、 r が、全ての g に対する \mathcal{R}_g に含まれる回数とする。

図 2 DD-ADMM による最適化アルゴリズム

ID の集合を表すと再定義する。 $G^{(t)} \subseteq G$ とし、 $G^{(t)}$ を $G^{(t)}$ の要素数とする。 $G^{(t)}$ は、 $\mathcal{R}^{(t)}$ に含まれる添字 ID の最適化変数で構成される部分問題の集合とする。また、記述を簡単にするため $\mathcal{Z}_g^{(t)} = \{z_{g,r}\}_{r \in \mathcal{R}_g, g \in G^{(t)}}$ を導入する。このとき、「動的変数整数計画問題系列」での時刻 t における整数計画問題 P_t は、双対分解により以下の最適化問題 L_t に変形できる:

$$\begin{aligned} (L_t) : \text{maximize}_{\mathcal{Z}_g^{(t)}, \mathcal{Z}^{(t)}} & \sum_{g \in G^{(t)}} \sum_{r \in \mathcal{R}_g} s_{g,r} z_{g,r} \\ \text{w.r.t. } & z_{g,r} \in \{0, 1\} \quad \forall z_{g,r} \in \mathcal{Z}_g^{(t)}, \\ & z_r \in \{0, 1\} \quad \forall z_r \in \mathcal{Z}^{(t)} \\ \text{s.t. } & \sum_{r \in \mathcal{R}_c} a_{c,r} z_{g,r} \leq b_c \quad \forall c \in \mathcal{C}_g, \forall g \in G^{(t)} \\ & z_{g,r} = z_r \quad \forall r \in \mathcal{R}_g, \forall g \in G^{(t)} \end{aligned} \quad (2)$$

全ての r に対するスコア関数 s_r が、任意の実数値定数 e に対して $e \sum_{g \in G} s_{g,r} = s_r$ という比例の関係が成り立つように各部分問題に対するスコア関数を設計することで、元の整数計画問題 P_t と、上記双対分解法に基づいて再定義した整数計画問題 L_t の最適解は一致する。つまり、この L_t を解けば元の整数計画問題 P_t の解が得られる。また、時刻 $T = 1$ とおけば、一般的な単一の整数計画問題と一致する。

4.2 双対分解法に基づく単一の問題に対する解法

実際に式 2 の時刻 t の整数計画問題 L_t を解く方法を述べる。まず、前述したように整数計画問題を直接解くのは一般的に計算量的な困難性があるため LP 緩和を用いる。次に、問題を解きやすい形に変形するとして、双対分解による定式化で導入した等式制約に対して拡張ラグランジュ

緩和法を適用しする。ラグランジュ乗数の集合を α とするとき、DD-ADMM[1, 2] の枠組みに基づいて、それぞれの最適化変数の集合 α , $z_g^{(t)}$, $z_r^{(t)}$ を反復計算により求める。図 2 に実際の計算アルゴリズムを示す。ただし、 k を繰り返しを管理する変数とし、 k 回目の繰り返し処理時の各変数を、それぞれ $\alpha_{g,r}^k$, $z_{g,r}^k$, z_r^k と記述する。同様に、それぞれの変数の集合も α^k , $z_g^{(t),k}$, $z_r^{(t),k}$ と書く。

4.3 動的変数整数計画問題系列への拡張

通常の整数計画法では、全ての r に関して $z_r = 0$ などで初期化して最適化を開始すればよい。しかし、本稿で考える動的変数整数計画問題系列では、時刻 $t-1$ の整数計画問題 P_{t-1} の最適解を得た最終状態を利用することで、時刻 t の最適解を効率よく求めたい。以下に、本稿での提案法を述べる。

4.3.1 前時刻 $t-1$ の最適化変数の状態を再利用

まず、もっとも単純な方法としては、時刻 $t-1$ の最適解を得た状態を時刻 t の初期値にして利用する方法が考えられる。時刻 $t-1$ と時刻 t の最適解が近ければ、この方法は直観的にうまく働くと考えられる。ただし、時刻変化によって変数や制約が増減しているので、単純に前時刻の最終状態を初期状態としても効果が有るかは必ずしも自明ではない。また、最適化アルゴリズムによっては、初期値があまり効果的に働かない場合もある。

しかし、本稿で利用する双対分解に基づく最適化アルゴリズムは、他の最適化アルゴリズムと比べ、各制約とその変数が他の制約やそこで扱われる変数と独立になるため、時刻 $t-1$ から時刻 t に変化した際に増減する部分問題の最適化変数 $z_{g,r}$ は他の変数との整合性等を考慮せず容易に追加/削除できるという利点がある。また、最適解を求めるアルゴリズムも変数を固定しての反復計算なので、徐々に最適解に近づくアルゴリズムである点も時刻 $t-1$ の状態を利用する効果が大いに期待できる。

よって、手続きとしては、時刻 $t-1$ から時刻 t に変化した問題 P_{t-1} から P_t に移行した際に、まず P_{t-1} との制約(部分問題)の観点での差分を取得する。このとき、追加に関しては $z_{g,r} = 0$, $\alpha_{g,r} = 0$ とし、新たに r が追加されたのであれば $z_r = 0$ とし、新たな最適化変数を生成する。逆に、削除の時は、 g に関する全ての $z_{g,r}$, $\alpha_{g,r}$ を削除し、削除した $z_{g,r}$ が最後の z_r に関する要素であれば、 z_r も削除する。これらの増減した部分問題以外の部分問題および等式制約の変数は、 P_{t-1} の最終状態を引き継ぐ。

4.3.2 活性/非活性状態制御

文献 [7, 1] では、最適化中に多くの最適化変数が一定の値で固定する状態が続く点に着目し、そういった変数を最適化から外すことにより最適化速度を向上させる方法を採用している。本稿では、こういった方法を更に発展させ、より緻密な最適化変数の状態制御の方法を提案する。結果として、提案する状態制御法は、本稿で取り上げる動的変数整数計画問題系列のような、最適化変数や制約が動的に変化する状況でその効果を発揮する方法である。

まず、最適化中の不要な計算を削減する仕組みとして、各部分問題 g と等式制約の変数 z_r 毎に、それぞれ「活性」「非活性」の二値の状態を持つフラグ F_g と F_r を付与する。「活性」状態 ($F_g = 1$ または $F_r = 1$) とは、 g または r は、通常の最適化処理を行うことを意味し、「非活性」状態 ($F_g = 0$ または $F_r = 0$) とは、 g または r は、最適化処理の計算から除外される事を意味する。

まず、図 2 にある最適化アルゴリズムは、図中の処理 1 および 2 と、3 および 4 という二種類の計算ブロックで表現でき、それぞれ一つの for 文の形で書き直すことができる。また、処理 1.2 に関しては、 z_r を定数として計算するので、各部分問題 g 毎に独立な変数 $z_{g,r}$ と $\alpha_{g,r}$ のみを変数となる計算となり、部分問題 g の単位で計算することができる。逆に、処理 3,4 に関しては、 $z_{g,r}$ は定数として計算するので、全て等式制約に基づく変数 z_r の単位で計算することができる。 F_g , F_r のフラグは以下の条件で

切り替わる。

- $F_g = 1 \rightarrow F_g = 0$
対応する部分問題 g に含まれる全ての変数 $z_{g,r}$ が $z_{g,r}^k = z_{g,r}^{k-1}$ となったとき、自身の部分問題 g は収束したと仮定。以降、 $F_g = 1$ が続く限り、部分問題 g に対する処理 1,2 は行われない。
- $F_r = 1 \rightarrow F_r = 0$
対応する r に対し $z_r^k = z_r^{k-1}$ となったとき、自身の変数 z_r は収束したと仮定。以降、 $F_r = 1$ が続く限り、 z_r に対する処理 3,4 は行われない。ただし、収束判定用の値はキャッシュしておき、非活性状態が続いても、キャッシュした値を用いることで適切に収束判定用の値を計算する。

これらの状態移行の条件は比較的緩めに設定してある。よって、かなりアグレッシブに本来収束していない場合も非活性状態へ移行する可能性がある。この問題を回避するために、非活性状態から活性状態に戻す処理を導入する。

- $F_g = 0 \rightarrow F_g = 1$
 $r \in R_g$ とする。制約条件 $z_r^{k-1} \neq z_r^k$ のとき、つまり部分問題 g にふくまれる最適化変数 $z_{g,r}$ と共有関係にある最適化変数 z_r が更新されたことを意味する。つまり、部分問題の解が最適化変数 z_r と一致していなかったことを示している。再度最適値を推定するために部分問題 g を活性状態に戻す。
- $F_r = 0 \rightarrow F_r = 1$
ある部分問題 g 中の $z_{g,r}$ が $z_{g,r}^{k-1} \neq z_{g,r}^k$ のとき、即ち、制約 z_r の値が変化することがあるので、再度最適解を評価するために活性状態にする。

ポイントは、フラグを非活性化するには、自分自身のフラグしか変更することができない。逆に、活性化するには、自分とは別カテゴリのフラグしか変更することができない、という設定になっている点である。これによって、状態制御がスタックすることが内容に設計されている。

このように非活性状態から活性状態へ変化させる処理を導入することで、動的変数整数計画問題系列の時刻変化による制約や最適化変数の増減に対応することが可能となる。つまり、容易に非活性状態から活性状態へ移行することが可能な枠組みを持つため、最適化変数の増減など周囲の状況が変化した場合でも、単純なフラグ管理で最小限の計算で最適化を行うことができる。

5 実験設定

本稿では、文章が時間と共に逐次挿入や削除されることにより変化していく状況を仮想的に再現し、その状況下で提案法とベースライン法とを比較し、提案法の効果を検証する。

5.1 タスク設定

実験データとして、英語 Penn Treebank のデータを利用する。係り受け解析などで用いられるように、WSJ セクション 02-21 を訓練用データ、セクション 22 を開発用データ、セクション 23 を評価用データとして用いる。

文章は一単語毎 (スペースキーまたは改行キー) が入力される、または、単語が削除される度に、その時点の文章に対して解析を行う設定とする。ただし、本稿の実験では、現時点では制御が困難な為、単語は必ず文章の最後に挿入されるか、文章の最後の単語が削除されることに限定する。文章編集の終了 ($t = T$) は、WSJ セクションの 1 ファイル分に相当する文章が全て入力された時点とする。よって、WSJ セクション 23(評価用データ) の場合は、丁度 100 文章 (2,416 文, 61,663 トークン) が評価用データになる。

自然言語解析タスクとしては、トークン分割、品詞付与、チャンキング、固有表現抽出、(部分) 係り受け、文分割の 6 種類のタスクを同時に解く問題とする。ただし、

表 1 実験の結果

| | トークン seg F. | 品詞 seg F. | チャンク seg F. | 係り受け ACC. | 固有表現 seg F. | 文分割 seg F. | 速度 (倍率) | 一括処理との 不一致数 |
|--------------|----------------|--------------|----------------|--------------|----------------|---------------|------------|----------------|
| 一括処理 $T = 1$ | 99.78 | 97.23 | 95.30 | 91.84 | 81.36 | 95.16 | 1 | — |
| P_t 毎に初期化 | 99.78 | 97.23 | 95.30 | 91.84 | 81.36 | 95.16 | 678.1 | 0/61663 |
| 決定的な解析 | 99.76 | 97.19 | 94.90 | 90.22 | 79.05 | 95.09 | 11.4 | 3994/61663 |
| 提案法 | 99.77 | 97.23 | 95.30 | 91.84 | 81.37 | 95.15 | 16.6 | 216/61663 |

係り受け解析に関しては、前述のように、一入力が一文章であるため、近距離の係り受け関係のみを対象とし、遠距離の係り受けに関しては、遠距離の何処にかかるというタグを推定する問題に置き換えた簡略化したものとする。正解データに関しては、トークン分割、品詞付与、文分割の正解データは Penn Treebank から、チャンキングの正解には CoNLL-2000 の正解作成スクリプトで生成されるチャンク、固有表現の正解は BBN データから、係り受けの正解データには Yamada's head rule により PTB の構文木を係り受け木に変換したものから、それぞれ抽出し結合したものの用いた。

5.2 モデル学習の設定

本稿では、自然言語を解析するときのみに着目しているため、事前に解析用のモデルは学習済みであることを想定している。実際のモデル学習には、個々のタスクに関してそれぞれ独立に online passive-aggressive 法 [8] を用いて学習を行った。

5.3 比較手法

本稿の実験では、文章が全て入力されてから解析を一回行う設定、つまり $T = 1$ の設定を最も基本的なベースラインとして用いる。次に、 T を単語を挿入/削除した回数に設定し、時刻 T での一致率と計算速度を比較する。

1. 一括処理: $T = 1$ と仮定し、全ての文章を一括で入力してから一回で最適化を行う方法
2. P_t 毎に初期化: 時刻 $t - 1$ の最終状態を用いずに、毎回全ての変数を 0 で初期化して最適化を行う方法
3. 決定的な解析: 時刻 $t - 1$ で得られた結果に関しては、そのまま固定し、毎時新たに追加された最適化変数のみを最適化する方法
4. 提案法: P_{t-1} の最終状態を初期値とし、更に活性/非活性状態も時刻 P_{t-1} の状態を引き継いで最適化を行う方法

ただし、速度比較の条件をそろえるために、単一の整数計画問題 P_t 内の最適化には、文献 [7, 1] のような収束した部分問題の計算をスキップして計算を高速化する方法を用いている。

5.4 結果および考察

表 1 に、実験結果を示す。トークン分割、品詞付与、チャンキング、固有表現、文分割は、セグメンテーション F 値 (Seg. F) を用いて解析精度を算出した。また、係り受け解析は、係り受け二項関係の正解率を用いた。

決定的な解析以外の結果は、ほぼ全て一致した。よって、解析精度の観点では、提案法、一括処理、毎回初期化する方法の間で差はない。決定的な方法は、特に係り受け解析の解析精度が大幅に落ち込んでいる。これは、決定的に解析を行うため、比較的長距離の係り受けは、後から入力される単語の情報により係り先が決定する場合があるため、このような結果になったと考えられる。逆に、提案法は、基本的に毎時与えられた全ての最適化変数を考慮して最適化を行うため、後から入力された単語に依存してこれまでの解析結果を変更する必要が生じても問題なく対応することができる。

計算速度に関しては、毎回初期化する場合と比較して、圧倒的に早く全体の最適化を行うことができている。逆に、決定的な方法よりも計算時間がかかっているのは、提案法は基本的に毎回全ての最適化変数を対象とした最適化を行うため、場合により最適化変数の数は数百倍以上となり、実質の計算コストは非常に大きい。ただし、活性/非活性制御の効果で、不要な計算コストを大幅に削減

することにより、決定的な方法と比較して約 1.6 倍程度の計算時間であった。計算時間に関しては、一括処理をするものが最も高速に計算できる。これは、毎回最適化を行う方法は、途中の時刻でも必ず最適解を求めているため余計な計算コストが発生しているためである。

今回の実験で最も注目したい重要な点は、全ての文章が入力されてから一括して一回最適化する場合と、提案法のように、途中一度も初期化せずに逐次最適化変数や制約が増減する環境で最適化を継続的に行った場合で最終状態が同じであれば、解は (ほぼ) 一致する点である。これは、文献 [3, 4, 5] でも言及されているように、最適化アルゴリズム ADMM の性質として、ネットワーク上での分散並列処理時に、任意のノードがダウンしたり、再起動したりすることで、変数が増減する環境でも十分な性能を発揮できることが知られている。提案法である活性/非活性状態制御法も変数の増減に対して高い適応力があり、かつ、最適化問題全体を効率よく計算できることがわかった。

6 まとめ

本稿では、時間に依存して解析対象となる文章が動的に変化する問題設定での自然言語解析問題を取り上げ、この問題設定を最適化変数や制約が離散的な時刻と共に動的に増減する整数計画問題の問題系列で定式化した。また、この問題に対して、文献 [1, 2] で提案された DD-ADMM を、部分問題と最適化変数を単位とした活性/非活性状態を制御して、問題系列全体を効率よく解くアルゴリズムを提案した。提案手法では、特に制約や最適化変数が時刻で増減する環境において、状態を保って継続して最適化を行ったとしても、最終状態が一致していれば、一括処理した場合と解析結果が (ほぼ) 一致することを示した。また、毎回初期化するよりも高速に計算でき、決定的な方法と比べても大幅に計算時間が増加しない方法であることを示した。将来、文章の編集途中でも、その時点での文章に対して最適な解析結果を与え、かつ、時々刻々と変化する文章に対して効率よく解析結果を提示する自然言語処理システムの構築に向けた指針を示した。

参考文献

- [1] Andre Martins, Noah Smith, Mario Figueiredo, and Pedro Aguiar. Dual decomposition with many overlapping components. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 238–249. Association for Computational Linguistics, July 2011.
- [2] André L. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. An augmented lagrangian approach to constrained map inference. In *ICML*, pages 169–176, 2011.
- [3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Foundations and Trends in Machine Learning, 2011.
- [4] Pedro Forero, Alfonso Cano, and Georgios Giannakis. Consensus-based distributed support vector machines. *JMLR*, 2010.
- [5] Kevin Duh, Jun Suzuki, and Masaaki Nagata. Distributed learning to rank on streaming data using alternating direction method of multipliers. In *Proc. of NIPS Big Learning Workshop (2011)*, 2011.
- [6] 鈴木 潤, Kevin Duh, and 永田 昌明. 拡張ラグランジュ緩和を用いた同時自然言語解析法. In 言語処理学会第 18 回年次大会, 2012.
- [7] Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October 2010. Association for Computational Linguistics.
- [8] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online Passive-Aggressive Algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.