情報処理学会論文誌 Vol.56 No.3 951–959 (Mar. 2015)

手書き変体仮名認識における単語の連接関係を用いた解の絞り込み

渡辺 悟[1,a)] 鈴木 徹也[1,b)] 相場 亮[1,c)]

受付日 2014年6月29日, 採録日 2014年11月10日

概要：古文書には変体仮名が使われており，その翻刻作業には多くの知識と労力を要する．それは変体仮名には異なる読みをする類似形状の文字や，同じ読みをする多様な形状の文字が存在するためである．以前我々の研究グループは古文書の翻刻支援を目的として制約充足による手書き変体仮名認識法を提案した．その手法では，各文字の読みの候補をあげ，読みの組合せの中から最適な単語列を探索する．この手法には，制約が少ないとき，組合せ爆発を起こすことがあった．本研究では，単語の生起コストと連接コストを導入し，コスト最小法と N-best 探索を用いて解の個数削減を図った．実験により従来手法の問題点を解決できたことを確認した．

キーワード：変体仮名，文字認識，N-best 探索，コスト最小法，制約解消

Reducing of the Number of Solutions Using Adjacency Relation of Words in Recognizing Historical KANA Texts

Satoru Watanabe[1,a)] Tetsuya Suzuki[1,b)] Akira Aiba[1,c)]

Received: June 29, 2014, Accepted: November 10, 2014

Abstract: It is difficult and time-consuming process to read Japanese historical manuscripts because historical KANA, which is a set of Japanese characters used in those manuscripts, contain different from those currently used. In our previous paper we formulated a part of the process, which recognizes historical KANA in a given historical manuscript, as a constraint satisfaction problem, and proposed a constraint solver. The solver enumerates current KANA possibly corresponding to historical KANA in a given historical manuscript, and searches the best combination of the enumerated current KANA where constraints are used to reduce search space. In our experiments, however, combinatorial explosion occurred when the number of constraints was not enough. In this paper we propose another method to assign current KANA to historical KANA in a given historical manuscript. Thanks to occurrence cost of words and connection cost of words, the method can limit the number of solutions using a minimum cost method and N-best search algorithm. Our experimental results shows our new method can avoid combinatorial explosion.

Keywords: historical KANA Texts, character recognition, N-best Search, minimum cost method, constraint solving

1. はじめに

古文書の翻刻は国文学の研究にとって基礎的な作業である．しかしそれには多くの知識と労力を必要とする．その理由の１つは，古文書に使われている変体仮名にある．現代仮名とは異なり，古典の平仮名においては同じ発音をする仮名に異なる形状の仮名文字が複数存在し，また異なる発音の仮名できわめて類似した形状を持つものも存在する．

そこで新井らは制約充足による手書き変体仮名認識システムの構想を提案した [1], [2]．そのシステムでは，まず画像認識器が入力画像中の各文字の読みの候補を列挙する．

1 芝浦工業大学大学院理工学研究科システム理工学専攻
Division of Systems Engineering and Science, Graduate School of Engineering and Science, Shibaura Institute of Technology, Saitama 337–8570, Japan
a) mf14061@shibaura-it.ac.jp
b) tetsuya@shibaura-it.ac.jp
c) aiba@shibaura-it.ac.jp

© 2015 Information Processing Society of Japan

次に制約解消器が，その読みの候補の組合せの中から全体が単語列となるような読みを求める．最後に，出力された複数の最適解の中から適切な読みを人間が選択する．

我々は，小規模な単語辞書を用いた場合でさえ，その制約解消器が組合せ爆発を抑えられない例を 8 章に示す実験により確認した．計算時間だけでなく出力する最適解の個数も膨大になるのが問題である．もし汎用的な単語辞書を用いれば考慮すべき組合せがさらに増えると予想される．

本論文では，上記の問題を解決するために，単語の連接関係を用いた制約解消器を新たに提案する．

以下，2 章では，翻刻と変体仮名について説明する．3 章では関連研究を説明し，その問題点を述べる．4 章で制約充足問題を説明し，続く 5 章で新井らが文献 [1] で提案した制約充足による手書き変体仮名認識の構想を述べる．6 章で新井らが実装した分枝限定法に基づく制約解消器とその問題点を説明する．その問題点を解決するために，7 章で新たな制約解消法を提案する．8 章では新井らの手法と提案手法との比較実験の結果とその評価を示す．最後の 9 章で，まとめと今後の課題を述べる．

2. 翻刻と変体仮名

翻刻とは，古文書などのくずし字で書かれた文章を楷書体に書き換えて読みやすくする作業であり，国文学の資料をテキストデータ化するために必要不可欠である．たとえば，**図 1** に示すような資料（伊勢物語 [3]）が翻刻の対象となる．翻刻には，くずし字の読解力，古典文法の知識，定型表現，文書の背景にある歴史学の知識などが必要である．そのため，翻刻は高度に専門的な作業である．

我々は，翻刻支援の第 1 歩として，漢字と比べて字種が少ない変体仮名の認識に取り組むことにした．

変体仮名（へんたいかな）とは，平仮名の字体のうち，1900 年の小学校令施行規則改正以降の学校教育で使用されていないものの総称である．異体字（いたいじ）とも呼ばれる．平仮名の元になった漢字を**字母**という．現在の平仮名は 46 種類であるが，字母の違いまで区別すると，古典の平仮名は 250 種類を超える．

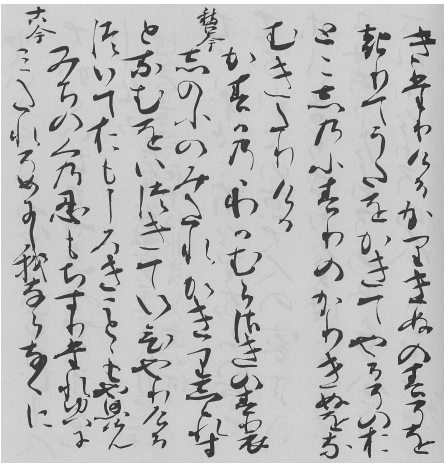

図 1 伊勢物語
Fig. 1 Tales of Ise.

古文書などでは**踊り字**も使用される．踊り字とは，直前の 1 文字または直前の 2 文字と同じ読みをする特殊な文字である．踊り字にはほとんど点に見えるもの，平仮名の「く」に字形が似ているものなどがある．

変体仮名に不慣れな者にとって，その読解は困難である．その理由の 1 つは，現代仮名に比べて，その字種が多いことである．一般に，日本語の 1 つの音を表す字母は複数存在し，1 つの字母にくずし方の異なる仮名が複数存在する．類似した形状を持つ仮名が異なる音を表す場合もある．さらに，毛筆で続けて書かれた文字について，1 文字の範囲を特定することも難しい．

3. 関連研究

手書き変体仮名認識の先行研究では，ニューラルネットワークを利用して文字単位で認識する和泉らの研究 [4] や，OCR と n-gram を併用して，前後の文字が既知の場合に不可読文字を推定する山田らの研究 [5] があり，どちらもあらかじめ用意した文字画像の特徴量を利用している．これらの研究は，文献を特定のものにすることにより，およそ 80〜90%の認識率を実現している．

和泉らの研究 [4] と山田らの研究 [5] は特定の文献に特化しているため汎用性に欠け，別の文献に対して使用する場合は画像特徴量を再び学習する必要がある．また，踊り字の認識は行われておらず，文字の並び方の利用も 2〜3 文字分と局所的である．

しかし一方では変体仮名の認識は人間でも困難であり，汎用化を目指せば，文字の形状のみでの認識では，認識率を上げることは困難であると考えられる．

現代日本語については，OCR の認識結果を統計的言語モデルを利用して誤り訂正をする研究 [6] や，形態素解析を利用した OCR ソフト [7] があるが，古典については，自然言語処理を利用した認識は見当たらない．

4. 制約充足問題

踊り字や単語となる文字の並び方，離れた文字の形の類似性を考慮するために，新井らは手書き変体仮名認識を制約充足問題として解くことにした．

制約充足問題（Constraint Satisfaction Problem, CSP）[8] とは与えられたすべての制約を満たすような各変数への値の割当てを求める問題であり，それは変数とその領域 $x_i \in D_i \quad (i = 1, \ldots, n)$ そして制約の集合で構成される．ここでいう**制約**とは変数が取り得る値の組の集合である．つまり，m 変数 $x_{i_1}, \ldots, x_{i_m}$ 間の制約を $C_{x_{i_1}, \ldots, x_{i_m}}$ とすると，$C_{x_{i_1}, \ldots, x_{i_m}} \subset D_{i_1} \times \ldots \times D_{i_m}$ である．変数 x_{i_j} $(j = 1, \ldots, m)$ に割り当てた値を d_j とし

© 2015 Information Processing Society of Japan

たとき，$(d_1, \ldots, d_m) \in C_{x_{i_1}, \ldots, x_{i_m}}$ ならば，その値は制約 $C_{x_{i_1}, \ldots, x_{i_m}}$ を満たすという．すべての制約を満たす値の組 $(d_1, \ldots, d_n) \in D_1 \times \ldots \times D_n$ をその CSP の**解**と呼ぶ．

CSP では制約どうしが競合し，すべての制約を同時に満たすことができない状況（**制約過多**）が生じる場合がある．一般的な CSP ではすべての制約を満たす値の組を解とするので，制約過多が生じた場合は解なしとなる．この問題を解決するために Borning らは制約に充足優先度を与え，**制約階層**を定義した [9]．優先度は自然数で表現される．その値が小さいほど充足優先度が高い．制約階層 $H = (H_0, \ldots, H_n)$ は，優先度ごとに分類した制約の集合の列である．ここで H_i は優先度 i を持つ制約の集合である．階層 H_0 に属する制約は**必須制約**と呼び，必ず満たさなければならない制約である．階層 $H_1, \ldots, H_n$ に属する制約は**選好制約**と呼び，満たされることが望まれる制約である．

Borning らは制約階層が導入された CSP の解の比較器も提案した [9]．ここでの解とは，変数に値を割り当てる写像のことである．2 つの解 θ，σ と制約階層 H に対して，解の比較器 **locally-predicate-better**（LPB）は次のように定義される [9]．

$$
\begin{aligned}
&\textit{locally-predicate-better}(\theta, \sigma, H) \\
&\quad \equiv \exists k > 0 \ s.t. \forall i \in 1 \ldots k-1 \ \forall p \in H_i \ e(p\theta) = e(p\sigma) \\
&\quad \wedge \exists q \in H_k \ e(q\theta) < e(q\sigma) \wedge \ \forall r \in H_k \ e(r\theta) \leq e(r\sigma)
\end{aligned}
$$

e は誤差関数である．$e(c\theta)$ は，解 θ が制約 c を満たすならば 0，満たさなければ 1 である．この比較器は解集合上に半順序を与える．解集合の中で，LPB で比較したときに極大な解を **locally-predicate-better（LPB）な解**という．

5. 制約充足による手書き変体仮名認識の構想

新井らが提案した制約充足による手書き変体仮名認識システムの構想 [1], [2] を**図 2** に示す．システムは古文書の画像を入力とし，翻刻結果を出力する．システムは画像認識器と制約解消器，そしてそれらが参照するデータベースとから構成される．画像認識器は，入力画像中の各文字の範囲を推定し，各文字の読みの候補を列挙する．その際，文字画像特徴量データベースを参照する．画像認識器は認識結果を CSP として記述し，それを制約解消器に渡す．制約解消器は，与えられた CSP を解き，適切な読みを各変数に割り当てる．このとき単語辞書を参照する．画像認識器は，その解を受け取り，各文字の読みの候補を修正する．画像認識器と制約解消器との間のやりとりを必要な回数だけ繰り返した後，システムは翻刻結果を出力する．

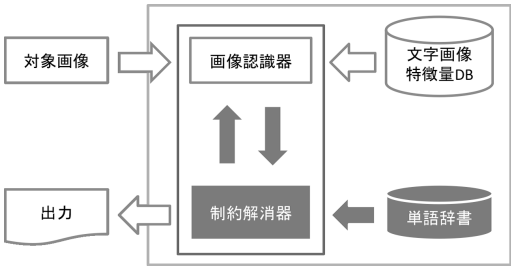

図 2 制約充足による手書き変体仮名認識システムの構成図
Fig. 2 A configuration diagram of our historical KANA Texts recognizing system using constraints.

この文字認識システムの技術的課題は画像認識器に関する課題と制約解消器に関する課題とに大きく分けられる．前者には画像からの文字の切り出し法や切り出した文字の認識方法などがある．後者には翻刻作業を CSP としてモデリングする方法や効率的な制約解消法などがある．

6. 分枝限定法に基づく制約解消器

新井らは Ruby 言語を用いて専用の制約解消器を実装した [1], [2]．この節ではその制約解消器の概要と問題点とを述べる．

6.1 翻刻制約充足問題

制約解消器は，翻刻に特化した CSP，**翻刻制約充足問題**（R-CSP）を扱う．それは，変数とその領域，制約の集合，読み順を与える有向非循環グラフ（DAG），そして単語辞書からなる．DAG の各ノードは変数に対応している．

R-CSP の変数は入力画像中の一文字を含む矩形に対応する．そして変数の領域はその矩形内の文字の読みの候補である．矩形どうしが重なることもありうる．

R-CSP の制約には，任意の優先度を持つ等号制約，非等号制約，単語開始制約，単語終了制約と変数に割り当てた読みが単語列になるという必須制約がある．たとえば，等号制約は，形状が類似する文字どうしが同じ読みをするという条件や，踊り字とその直前の文字とが同じ読みをするという条件を表すのに用いる．非等号制約は，形状が異なる文字どうしが違う読みをするという条件や，各文字が不可読文字ではないという条件を表すのに用いる．不可読文字とは，システムの画像認識器が正しく文字を認識できなかったことを表す特別な文字である．

6.2 制約解消手順の概要

変数に割り当てた読みが単語列になるという制約は DAG に次のようなパスが存在するという条件である．

- ある最上流ノードからある最下流ノードへ至るパスによって得られる変数列を $x_1, \ldots, x_n$ とする．
- その変数列中の各変数 $x_1, \ldots, x_n$ に読みを割り当てて得られる記号列を $r_1 \ldots r_n$ とする．
- $r_1 \ldots r_n$ が単語辞書に載っている単語の列である．ただし最初と最後の単語は途中で切れていても構わない．

DAG 上で上記のようなパスを 1 つ選ぶと，全体の読みが 1 つ定まる．このとき選択したパス上にない変数には読

© 2015 Information Processing Society of Japan

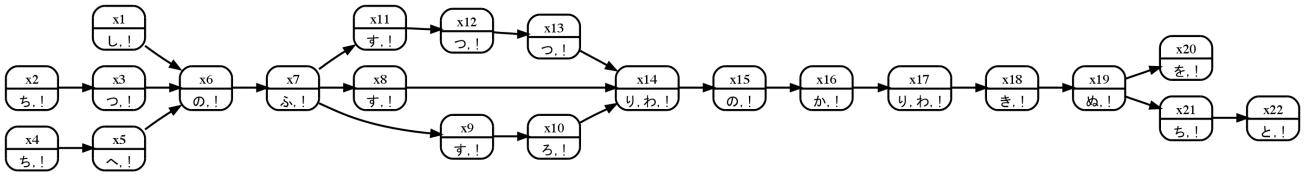

図 3　R-CSP を表す制約グラフの例
Fig. 3　A constraint graph for a R-CSP.

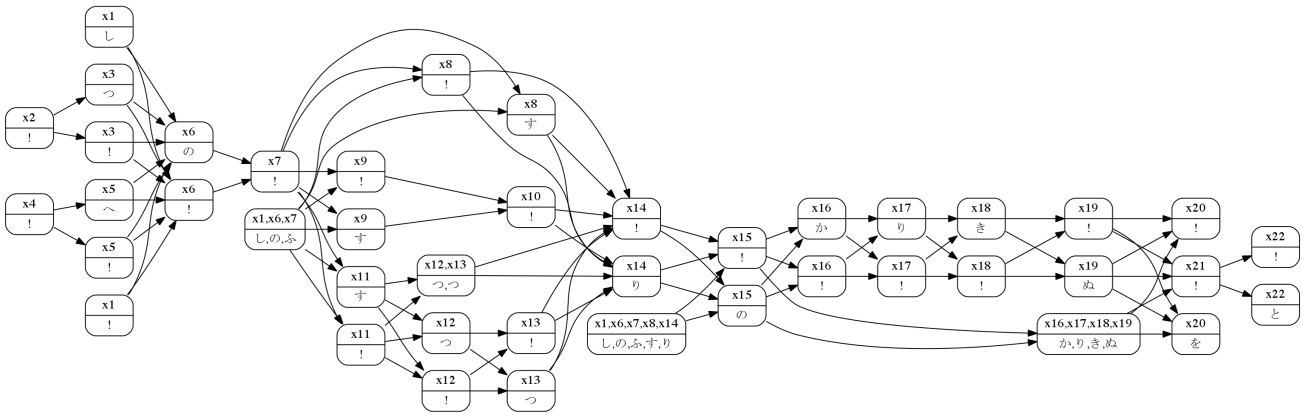

図 4　図 3 のグラフを元に構築した読みの割当てグラフ
Fig. 4　A reading-assignment graph built from the graph of Fig. 3.

みを割り当てないことにする．そして，2 つの解を LPB で比較する際，読みが割り当てられていない変数に係わる制約は満たされていると見なすことにする．

制約解消器は，次の手順に従い，R-CSP の LPB な解をすべて求める．

(1) R-CSP を表す DAG（**制約グラフ**）を元に，辞書を用いて可能な読みをすべて割り当てたグラフ（**読みの割当てグラフ**）を作成する．
(2) 分枝限定法を用いて LPB な解をすべて探索する．

制約グラフの例を**図 3** に示す．各ノードの上半分に変数，下半分にその領域が記されている．各ノード間の有向辺は読みの順番を表している．入力画像中の 1 文字分の矩形内に別の矩形が存在すると，制約グラフ上で分岐や合流が現れる．たとえば，図 3 の右端に分岐が存在するのは，変数 x_{20} に対応する矩形内に 2 変数 x_{21}，x_{22} に対応する 2 つの矩形が存在するからである．このグラフを元に，単語辞書を参照しながら，読みを割り当てることで読みの割当てグラフを作成する．**図 4** がその例である．各ノードの上半分が変数列，下半分がその変数列に割り当てられた文字（単語）である．読みの割当てグラフを作成することで，最上流から最下流へたどるすべてのパスが単語の列となる．

手順 (2) では，手順 (1) で作成した読みの割当てグラフの最上流ノードから下流に向かってノードを訪問し，分枝限定法を用いて LPB な解の探索を行う．

6.3　制約解消器の問題点

8 章の比較実験で示すように，240 語程度の小規模な単語辞書を使った場合でさえ，新井らの制約解消器は組合せ爆発を抑えられないことがある．汎用的な単語辞書を用いればノード数は増え，解の数と計算時間もさらに増加すると考えられる．制約を多くすれば組合せ爆発を抑えられるかもしれないが，制約を自動で加えることは難しい．

計算時間の増加の解決方法として，Sugar [10] のような既存の高速な制約解消器で R-CSP を解くことが考えられる．しかし，R-CSP では単語辞書を参照する必要があるので，そのためには単語辞書を一般的な制約で記述する必要がある．それでは制約の量が膨大になり，効率が悪くなることが予想される．読みの割当てを行ったあとに既存の高速な制約解消器で扱える形に変換して解くことも考えられるが，これらの方法で計算時間の増加を抑えられたとしても，解の数の増加は抑えられない．

7.　単語の連接関係を利用する制約解消器

6 章で述べた問題点である組合せ爆発を防ぐため，単語の生起コストと連接コストを導入し，コスト最小法と N-best 探索を用いて解の個数を削減することを提案する．R-CSP では変数に割り当てた読みが単語列になるので，コスト最小法のような単語の連接関係を利用した手法を活用することができる．

7.1　コスト最小法と N-best 探索

コスト最小法とは，形態素解析に用いられる手法の 1 つである [11]．コスト最小法は，単語の生起コストと連接コストを定義し，コストの総和が最小となるパスを最適解と

© 2015 Information Processing Society of Japan

して選択する方法である．生起コストはノードに与えられ，その単語が表れにくいほど高くなるようにする．連接コストはリンクに与えられ，両端のノードの単語が連接しにくいほど高くなるようにする．

また，一般に上位 N 個の最適解を求めることを **N-best 探索**という [12]．コスト最小法における上位 N 個の最適解は，前向き動的計画法と後向き A*アルゴリズムの組合せによって効率良く求めることができる．

7.2　制約解消手順

提案手法の手順を以下に示す．手順 (1) は新井らの手法と同じである．

(1) R-CSP から読みの割当てグラフを作成する．
(2) 読みの割当てグラフからコスト付き読みの割当てグラフを構築する．
(3) コストの総和が少ない順に上位 N 個の解を求める．
(4) 得られた解の中から LPB な解を残す．

コスト最小法によってつながりやすい単語列を選択することで，良い解が得られることが期待できる．N-best 探索によって解の数を任意に調整することができ，問題のサイズや辞書のサイズが大きくなっても計算時間を抑えることができる．

以降，手順 (2) と手順 (3) を順に説明する．

7.3　コスト付き読みの割当てグラフの構築

手順 (2) で構築する**コスト付き読みの割当てグラフ**とは，読みの割当てグラフに対して以下の拡張を行ったグラフである．

- 新井らの手法では単語の種類（品詞など）を区別していないので，それを区別する．
- 文頭であることを表す文頭ノードと，文末であることを表す文末ノードを追加する．
- ノードに生起コスト，リンクに連接コストを与える．

文頭ノードと文末ノードは，ある単語が先頭か末尾に現れるコストを考えるために追加する．新井らの手法ではパスの先頭と末尾が単語の途中であることを許しているが，本手法では単語の途中であるノードは，連接コストを付与するときは文頭ノードや文末ノードとして扱い，生起コストは別に定義して付与する．

コスト付き読みの割当てグラフの構築には，コスト付き単語辞書を使用する．**コスト付き単語辞書**は手順 (1) で使用する単語辞書とは違い，単語の種類が区別されており，単語の生起コストと連接コストの情報を持っている辞書である．コスト付き読みの割当てグラフの構築手順を以下に示す．

(1) 読みの割当てグラフの各ノードについて，コスト付き単語辞書を引き，コスト付き読みの割当てグラフのノードを作成する．このとき，もとの読みの割当てグラフのノードへの対応付けを記録する．
(2) コスト付き読みの割当てグラフの各ノードについて，対応する読みの割当てグラフのノードの下流ノードに対応する各ノードに対してリンクを張り，連接コストを付与する．
(3) 文頭ノードを作成し，文頭ノードから読みの割当てグラフの最上流ノードに対応する各ノードに対してリンクを張り，連接コストを付与する．
(4) 文末ノードを作成し，読みの割当てグラフの最下流ノードに対応する各ノードから文末ノードに対してリンクを張り，連接コストを付与する．

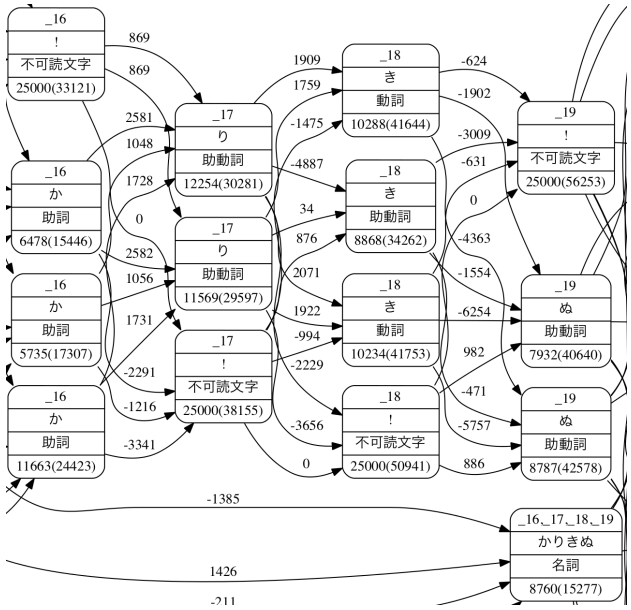

図 5　図 4 の読みの割当てグラフをもとに構築したコスト付き読みの割当てグラフの一部

Fig. 5　A part of a reading-assignment graph with costs built from the graph of Fig. 4.

図 4 の読みの割当てグラフをもとに構築したコスト付き読みの割当てグラフの一部を**図 5** に示す．この図では文頭ノードから各ノードまでの最小コストはすでに計算されている．リンクに付与されている数字は連接コストである．ノードが持っている情報は上から順に，変数名，割り当てられた読み，品詞，生起コスト（文頭ノードからそのノードまでの最小コスト）である．「か，助詞」など読みと品詞が同じノードが 2 つ以上存在するのは，この図のグラフを構築する際に単語が品詞以上（係助詞，終助詞など）に細かく分類されている辞書を使用したからである．

7.4　読みの N-best 探索

手順 (3) は前向き探索と後ろ向き探索との 2 段階で構成される．

7.4.1　前向き探索

まず，手順 (2) で構築したコスト付き読みの割当てグラフの文頭ノードから各ノードまでの最小コストを動的計画

© 2015 Information Processing Society of Japan

法で求める．最小コストは，トポロジカル順序で上流から下流に向かってノードを取り出し，各下流ノードの対して最小コストを更新していくことで求まる．そのアルゴリズムとそこで用いられるコストの記法の定義を以下に示す．

$\boldsymbol{h(n)}$ 文頭ノードからノード n までの最小コスト

$\boldsymbol{c(n)}$ ノード n の生起コスト

$\boldsymbol{cost(n, m)}$ 上流ノード n と下流ノード m の連接コスト

(1) 文頭ノード i について $h(i) := 0$，それ以外の全ノード o について $h(o) := \infty$ とする．

(2) トポロジカル順序でノード n を取り出す．取り出せなければ終了する．

(3) n の各下流ノード m に対して以下の操作を行う．

(a) $h' := h(n) + cost(n, m) + c(n)$ とする．

(b) $h(m) > h'$ のとき，$h(m) := h'$ とする．

(4) (2) に戻る．

7.4.2 後ろ向き探索

最小コストが求まったら，文末ノードから A*アルゴリズムを応用して探索を行い，上位 N 個の解を求める．ここで新たに 2 種類のコストの記法を以下のように定義する．

$\boldsymbol{f(n)}$ 文末ノードからノード n を通って文頭ノードに到達するときの最小コスト

$\boldsymbol{g(n)}$ 文末ノードからノード n までの最小コスト

すると 4 種類のコストの間に $f(n) = g(n) + h(n) - c(n)$ が成り立つ．$h(n)$ は前段階で求まっているので，$f(n)$ が最小のノードから探索していけば，最小コストのパスがただちに求まる．

通常の A*アルゴリズムでは最小コストの解だけを求めればよいので，コスト付き読みの割当てグラフ上で探索を行い，最小コストの比較をして最小コストと親の更新を行うことで，無駄な計算をしないようにする．しかし，N-best 探索を行うときは，すべての組合せを考える必要があるので，コスト付き読みの割当てグラフとは別に探索木を考え，総当たりになるようにノードを展開する．

そのアルゴリズムを以下に示す．また，探索木のノードを探索ノードと呼ぶ．探索ノードにはコスト付き読みの割当てグラフのノードの 1 つが関連付けられており，探索ノード x に関連付けられているコスト付き読みの割当てグラフのノードを $n(x)$ とする．$f(x)$，$g(x)$ は探索ノード x についてのコストとなる．

(1) 探索ノードのリスト q を空にする．

(2) 探索ノード t を q に加える．$n(t) :=$ 文末ノード，$f(t) := h(n(t))$ とする．

(3) q が空，または得た解の個数が N なら探索を終了する．

(4) q から最小の $f(x)$ を持つ探索ノード x を取り出す．

(5) $g(x) := f(x) - h(n(x)) + c(n(x))$ とする．

(6) $n(x)$ が文頭ノードのとき x の親をたどり解をえて (3) に戻る．

(7) $n(x)$ の各上流ノード m に対して以下の操作を行う．

(a) 探索ノード y を用意する．

(b) $n(y) := m$ とする．y の親を x にする．

(c) $f(y) := g(x) + h(m) + cost(m, n(x))$ とする．

(d) y を q に加える．

(8) (3) に戻る．

7.4.3 動作例

例を使用して N-best 探索の後ろ向き探索の部分について説明する．使用するグラフを**図 6** に示す．アルファベットはノード名，数字はそれぞれノード，リンクに付与されている生起コスト $c(n)$，連接コスト $cost(n, m)$ である．括弧内の数字はノード S からそのノードまでの最小コスト $h(n)$ である．S は文頭ノード，G は文末ノードにあたる．

後ろ向き探索の探索木を**図 7** に示す．ノードの上の数字は $f(x)$，図の最下部の数字はそのノードの親をたどって得られるパスのコストの総和である．

探索ノードのリスト q の内容の変化を追うことにする．リストの内容は [ノード名（そのノードの $f(x)$ の値), ...] と表記する．コスト付き読みの割当てグラフのノード X に対応する探索木のノードを X' と表記する．最初，q は $[G'(43)]$ となっている．q から $G'(43)$ が取り出され，G の上流ノードである E，F に対応する $E'(43)$，$F'(55)$ が q に追加される．q は $[E'(43), F'(55)]$ となる．次に最小コストを持つ $E'(43)$ が取り出される．E の上流ノードである C，D に関して，$f(C') := (43 - 38 + 10) + 21 + 7 = 43$，$f(D') := (43 - 38 + 10) + 26 + 6 = 47$ となり，$C'(43)$，$D'(47)$ が q に追加される．q は $[C'(43), D'(47), F'(55)]$ となる．その後，$[A'(43), B'(45), D'(47), F'(55)]$，$[S'(43),$

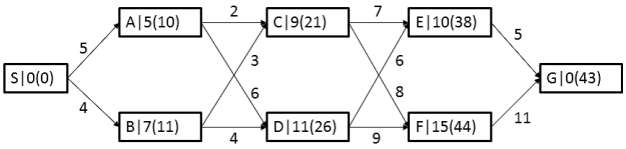

図 6 簡略化したコスト付き読みの割当てグラフの例

Fig. 6 A simplified reading-assignment graph with costs.

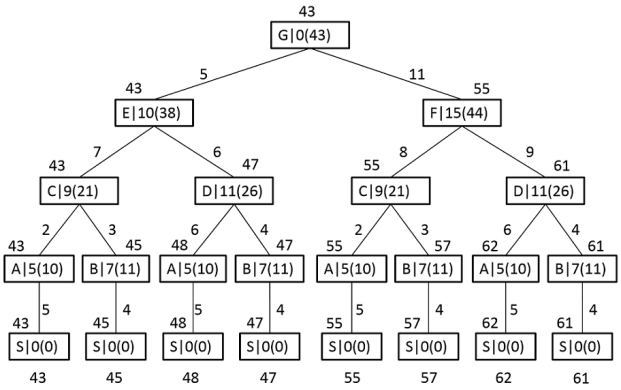

図 7 図 6 のグラフ上での後ろ向き探索の探索木

Fig. 7 The search tree of backward search on the graph of Fig. 6.

© 2015 Information Processing Society of Japan

表 1 実験に使用する R-CSP の大きさと正解の文字列長
Table 1 Size of R-CSPs and lengths of correct strings used for experiment.

R-CSP	変数の数	制約の数	正解の文字列長
ise2_1	71	72	47
ise2_2	100	108	59
ise2_3	30	30	28
ise2	201	210	134
ise2_1add1	71	72	47
ise2_1add2	71	72	47
ise2_1add3	71	72	47
ise2_1add3_c1	71	78	47
ise2_1add3_c2	71	85	47
ise2_1add3_c3	71	88	47

$B'(45), D'(47), F'(55)]$ となり，文頭ノード $S'(43)$ が取り出されるので，コストが最小である解 $SACEG$ が得られる．さらにその後，最小コストを持つ $B'(45)$ が取り出され，$S'(45)$ が q に追加される．このように探索を続けていくと，コストが少ない順に解を得ることができる．

8. 実験

本章では，7 章で述べた提案手法による解の削減の実験について説明し，その結果をまとめる．

8.1 目的と方法

新井らの手法 [1], [2] と提案手法との比較，辞書サイズの大きさによる比較，読みの候補数による比較，制約の数による比較のために実験を行った．対象は伊勢物語 [3] の一部とし，R-CSP は図 1 から人手で作成した．コスト付き単語辞書は形態素解析辞書である中古和文 UniDic [13] から作成した．使用した計算機の CPU は Intel Core i5，クロック数 1.8 GHz，搭載メモリ 8 GB である．プログラムは Ruby 言語で実装した．

8.2 対象の R-CSP

実験に使用する R-CSP の大きさと正解の文字列長を**表 1** に示す．

図 1 から作成した R-CSP が ise2 であり，ise2_1，ise2_2，ise2_3 は ise2 を分割したものである．これらは，新井らの手法で解を得るためと，R-CSP の正解の文字列長に対して計算時間がどう変化するか調べるために作成した．

名前に add とついていない R-CSP の各文字の読みの候補はたかだか 3 つである．add とついている R-CSP は，ise2_1 の正解のパスに含まれているすべての変数の読みの候補を，add1 は 2 から 3 つ，add2 は 3 つ，add3 は 4 つに増やしたものである．これらの R-CSP は読みの候補数による比較を行うために作成した．

ise2_1add3_c1，ise2_1add3_c2，ise2_1add3_c3 以外の R-CSP が含む制約は次の 2 種類である．1 つ目は，すべての変数は不可読文字ではないという非等号制約である．2 つ目は，踊り字に対応する変数は，その直前の文字に対応する変数と同じ読みをするという等号制約である．不可読文字はすべての変数の読みの候補に含まれている．不可読文字ではないという非等号制約は選好制約であり，どのような読みの組合せも単語列にならない場合に，不可読文字は解に含まれることがある．

ise2_1add3_c1 は，ise2_1add3 に等号制約を 1 つ，単語開始制約を 1 つ，単語終了制約を 4 つ追加したものである．ise2_1add3_c2 は，それに加えて非等号制約を 7 つ，ise2_1add3_c3 はさらに等号制約を 3 つ追加したものである．等号制約は図 1 の最初の文字に対応する変数と，最初の文字と同じ読みをする文字に対応する変数に加えた．非等号制約は最初の文字に対応する変数と，最初から 5 番目の文字までに対応する変数に加えた．単語開始制約は最初の文字に対応する変数に加えた．単語終了制約は最後の文字に対応する変数に加えた．これらの R-CSP は制約の数による比較を行うために作成した．

表 2 実験に使用する単語辞書
Table 2 Word dictionaries used for experiment.

辞書	語数	コスト付き単語辞書の語数
ise2dic	241	363
unidic	257,879	421,896

8.3 単語辞書

実験に使用する単語辞書の特徴を**表 2** に示す．辞書 ise2dic は対象に含まれている単語と基本的な助詞，助動詞からなっている．辞書 unidic は中古和文 UniDic 中の漢字仮名混じりの単語を平仮名に変え，濁点と半濁点を除いた辞書である．

2 つの辞書 ise2dic と unidic と同じ単語からなるコスト付き単語辞書をそれぞれ作成した．ただし，コスト付き単語辞書では，品詞以上に細かく単語を分類している．ise2dic には中古和文 UniDic にない単語も含まれており，それらの単語のコストは類似単語のコストで代用した．

8.4 結果

8.4.1 新井らの手法と提案手法との比較

新井らの手法と提案手法による ise2dic を用いた実験結果をそれぞれ**表 3**，**表 4** に示す．30 分以内に計算が終わらなかったとき，測定不能とした．また，計算時間に読みの割当てグラフの作成は含まれていない．ノード数は，新井らの手法の実験結果では読みの割当てグラフのノード数，提案手法の実験結果ではコスト付き読みの割当てグラフのノード数である．表 4 のノード数が表 3 より増加しているのは，単語の種類を区別したためである．平均正解

© 2015 Information Processing Society of Japan

表 3 新井らの手法の実験結果（ise2dic）
Table 3 Experimental results by our previous method (ise2dic).

R-CSP	ノード数	解の数	平均正解率	計算時間
ise2_1	152	41,472	0.922	272 秒
ise2_2	230	測定不能		
ise2_3	72	28	0.870	0.13 秒
ise2	449	測定不能		

表 4 提案手法の実験結果（ise2dic，N = 10）
Table 4 Experimental results by our new method (ise2dic, N = 10).

R-CSP	ノード数	解の数	平均正解率	計算時間
ise2_1	268	10	1.000	0.18 秒
ise2_2	402	10	0.953	0.20 秒
ise2_3	144	10	0.932	0.16 秒
ise2	805	10	0.962	0.34 秒

表 5 提案手法の実験結果（unidic，N = 10）
Table 5 Experimental results by our new method (unidic, N = 10).

R-CSP	ノード数	解の数	平均正解率	計算時間
ise2_1	2,237	10	0.972	0.77 秒
ise2_2	4,053	6	0.938	1.16 秒
ise2_3	1,262	10	0.961	0.66 秒
ise2	7,567	6	0.958	1.89 秒

率は正解した文字数を正解の文字列長で割ったものの平均である．

表 3，表 4 からほとんどの場合で新井らの手法より解の数，計算時間が少なく，より良い解が得られていることが分かる．

ise2_1 について比較すると，新井らの手法より解の数は 4,147 分の 1，計算時間は 1,511 分の 1 までに減少し，平均正解率は 0.922 からすべて正解である 1.0 に上昇している．

8.4.2 辞書の変化による比較

辞書を unidic に変更して行った実験結果を**表 5** に示す．表 4，表 5 から，辞書の変更により ise2dic の実験結果と比べて，ノード数は約 8 から 10 倍，計算時間は約 4 から 5.5 倍に増加していることが分かる．ise2_3 以外では平均正解率は減少している．

unidic の辞書データ読み込みには 0.37 秒かかっている．この処理時間を表 5 の計算時間から引くと，ise2 は 1.52 秒，ise2 を分割したものである ise2_1，ise2_2，ise2_3 の合計は 1.48 秒となる．

8.4.3 読みの候補数による比較

ise2_1 について各変数の読みの候補を増やして行った実験結果を**表 6** に示す．表 6 から，読みの候補の増加にともない，ノード数と計算時間は増加し，平均正解率は減少していることが分かる．ise2_1add3 の変数の数は，変数の数が最も多い ise2 の 3 分の 1 程度（表 1）だが，ノード数は同条件（unidic，N = 10）の ise2 の 7,567（表 5）よりも多く，計算時間も 4 倍以上になっている．また，ise2_1add3 の平均正解率は ise2_1add2 以外と比べて 0.3 以上減少している．

表 6 提案手法での読みの候補数を変化させた実験結果（unidic, N = 10）
Table 6 Experimental results changing the number of candidates of reading by our new method (unidic, N = 10).

R-CSP	ノード数	解の数	平均正解率	計算時間
ise2_1	2,237	10	0.972	0.77 秒
ise2_1add1	3,391	10	0.932	1.20 秒
ise2_1add2	6,323	10	0.743	2.86 秒
ise2_1add3	10,997	10	0.617	8.46 秒

表 7 提案手法での制約の数を変化させた実験結果（unidic, N = 500）
Table 7 Experimental results changing the number of constraints by our new method (unidic, N = 500).

R-CSP	制約の数	解の数	平均正解率	計算時間
ise2_1add3	72	500	0.602	15.74 秒
ise2_1add3_c1	78	165	0.695	15.94 秒
ise2_1add3_c2	85	26	0.713	15.99 秒
ise2_1add3_c3	88	9	0.735	16.13 秒

8.4.4 制約の数による比較

ise2_3add3 について制約の数を追加して行った実験結果を**表 7** に示す．表 7 から，制約数の増加にともない，解の数は減少し，平均正解率と計算時間は増加していることが分かる．ise2_1add3_c3 では ise2_1add3 と比べて，解の数は 55 分の 1 に減少し，平均正解率は 0.133 増加し，計算時間は約 2.5%増加した．

8.5 評価

新井らの手法と提案手法との比較により，新井らの手法の問題点である解の数と計算時間を大幅に減少させられたことが確認できた．また，平均正解率の向上も見込めた．

辞書の変化による比較から，汎用的な辞書である unidic を使用しても，平均正解率はそれほど減少しないことが分かったので，限定的な辞書から汎用的な辞書に変更しても問題はないと考えられる．

辞書データ読み込みの時間を省いて考えると，ise2 は 1.52 秒，ise2_1，ise2_2，ise2_3 の合計は 1.48 秒となっている．正解の文字列長に対して大体線形時間であると考えられる．

読みの候補数による比較では，読みの候補が 4 つで平均正解率は 0.617 まで減少した．したがって，実用には精度の高い画像認識器が必要だと考えられる．さらに，計算時間も他の実験結果より大きくなっており，読みの候補数増加による影響は他の変化よりも大きいと考えられる．

© 2015 Information Processing Society of Japan

制約の数による比較では，制約を追加しても計算時間をほとんど増加させることなく，解の数を絞り込んで，平均正解率の高い解を得ることができた．これはこの実験では正解が満たしている条件を制約として追加したためである．しかし解の数に影響しない制約や正解とは相反する条件となる制約もありうるので，一般には制約を追加してもこのような傾向が得られるとは限らない．

9. おわりに

本論文では，手書き変体仮名認識について，単語の生起コストと連接コストを導入し，コスト最小法と N-best 探索を用いて解の個数を削減することを提案した．新井らの手法 [1], [2] より解の数と計算時間が少なく，より良い解が得られたことを実験により確認した．また，辞書や問題のサイズが増えても，現実的な計算時間で解くことができることを実験により確認した．しかし，読みの候補が増加すると良い解が得られなくなってしまうことが分かった．

現在，R-CSP の作成は人手で行われているが，読みの候補を 2 つから 3 つほどに絞れる画像認識器と R-CSP の自動作成ができれば，実用的な手書き変体仮名認識システムになると考えられる．

今後の課題として，単語辞書にない単語の考慮や，画像特徴量による文字認識結果の順位の利用検討があげられる．また，制約充足による手書き変体仮名認識システム全体の課題としては，画像認識器と制約解消器の連携方法や，R-CSP の自動作成と作成支援，画像認識器の精度向上があげられる．

参考文献

[1] 新井侑太，鈴木徹也，相場　亮：手書き変体仮名認識における制約充足問題の拡張，第 75 回全国大会講演論文集，pp.331–333 (2013).

[2] Arai, Y., Suzuki, T. and Aiba, A.: Recognizing Historical KANA Texts using Constraints, *Theory and Practice of Computation*, Vol.7, pp.151–164 (2013).

[3] 鈴木知太郎：御所本伊勢物語 冷泉為和筆 宮内庁書陵部蔵影印本，笠間書院 (1994).

[4] 和泉勇治，加藤　寧，根元義章，山田奨治，柴山　守，川口洋：ニューラルネットワークを用いた古文書個別文字認識に関する一検討，情報処理学会研究報告，人文科学とコンピュータ研究会報告，Vol.2000, No.8, pp.9–15（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110002930070/⟩（参照 2000-01-21）.

[5] 山田奨治，柴山　守：n-gram と OCR による定型表現がある古文書の文字の推定，技術報告 59 (2003-CH-058)，国際日本文化研究センター・研究部，大阪市立大学・学術情報総合センター (2003).

[6] 竹内孔一，松本裕治：統計的形態素解析と文字 n-gram を利用した OCR 誤り訂正，情報処理学会研究報告，自然言語処理研究会報告，Vol.99, No.22, pp.17–24（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110002935085/⟩ (1999).

[7] オムロンソフトウェア株式会社：組み込み型日本語形態素解析 SuperMorpho-J，オムロンソフトウェア株式会社（オンライン），入手先 ⟨http://www.omronsoft.co.jp/SP/embedded/morpho/jirei.html⟩（参照 2014-09-11）.

[8] Apt, K.: *Principles of Constraint Programming*, Cambridge University Press, New York, NY, USA (2003).

[9] Borning, A., Feldman-Benson, B. and Wilson, M.: Constraint Hierarchies, *LISP AND SYMBOLIC COMPUTATION*, pp.48–60 (1992).

[10] 田村直之：Sugar: A SAT-based Constraint Solver, 神戸大学（オンライン），入手先 ⟨http://bach.istc.kobe-u.ac.jp/sugar/⟩（参照 2014-09-09）.

[11] 奥村　学：自然言語処理の基礎，コロナ社 (2010).

[12] 永田昌明：統計的言語モデルと N-best 探索を用いた日本語形態素解析法，情報処理学会論文誌，Vol.40, No.9, pp.3420–3431（オンライン），入手先 ⟨http://ci.nii.ac.jp/naid/110002725063/⟩(1999).

[13] 国立国語研究所：中古和文 UniDic ver.1.3，国立国語研究所（オンライン），入手先 ⟨http://www2.ninjal.ac.jp/lrc/index.php?UniDic%2F%C3%E6%B8%C5%CF%C2%CA%B8UniDic⟩（参照 2014-01-08）.

渡辺 悟　（学生会員）

2014 年芝浦工業大学システム理工学部電子情報システム学科卒業．現在，芝浦工業大学大学院理工学研究科システム理工学専攻修士課程に在籍．

鈴木 徹也　（正会員）

2000 年東京工業大学大学院情報理工学研究科計算工学専攻博士課程修了．博士（工学）．現在，芝浦工業大学大学院理工学研究科システム理工学専攻准教授．

相場 亮　（正会員）

1986 年慶應義塾大学院工学研究科数理工学専攻博士課程後期修了．工学博士．現在，芝浦工業大学大学院理工学研究科システム理工学専攻教授．

© 2015 Information Processing Society of Japan