# 単語長を考慮した最長しりとり問題の実験的考察

乾 伸雄[†] 品野勇治[†] 小谷善行[†]

本論文では，先行研究の最長しりとり問題の一般化として，最長しりとり問題で単語の長さを考慮した文字数最大しりとり問題の厳密解法について述べ，その実験的評価を行う．文字数最大しりとり問題は，整数計画問題として記述した場合，単語の最大の長さを $l$ としたとき，最長しりとり問題を記述するための変数の $l$ 倍の変数が必要となり，現実的に解けるかどうかは未知である．$l = 26$ の既知の単語について，文字数最大しりとり問題は先行研究での最長しりとり問題に比べ，約 41 倍の計算時間がかかることが分かった．さらに，これら 2 つの問題の派生問題として，固定単語長文字数最大しりとり問題および固定文字数最長しりとり問題を取り上げる．これらの派生問題を用いて，広辞苑と ICOT 形態素解析辞書について，最長しりとり問題の最適解と文字数最大しりとり問題の最適解の関係を分析した．

# Experiment on the Maximum-Shiritori-String-Length Shiritori Problem

Nobuo Inui,[†] Yuji Shinano[†] and Yoshiyuki Kotani[†]

This paper describes an exact algorithm of the maximum-shiritori-string-length shiritori problem (MS3 in short) which maximizes a shiritori-string length. This algorithm is obtained from a generalization of an exact algorithm of the longest shiritori problem (LS in short) proposed previously. We experimentally evaluate the algorithm and investigate the properties of the MS3 problem. Since the MS3 problem takes $l$ times number of variables of the LS problem, where $l$ is the maximum length of words, under the integer programming approach, it is unknown whether the problem instance can be solved or not. Our experimental results showed that 41 times calculation times of the LS problem is required to solve MS3 problem, when $l = 26$. In addition, two derived problem of these shiritori problems, called the fixed-length MS3 shiritori problem and the fixed-shiritori-string-length LS problem, are introduced. In this paper, we analyze the relations between the MS3 problem and the LS problem, using these derived problems.

## 1. はじめに

しりとりは，単語の末尾の音に対して同じ音で始まる単語をつないでいき，次の単語を出すことができなかったゲーム参加者が負けとなるゲームである．しりとりは言葉遊びとして日常的なものである．コンピュータでしりとりを扱った研究には，連想しりとりの研究[2] や 2 人完全情報ゲームとしてしりとりを扱った研究[1]，また，最長しりとり問題[7] を解く研究がある．最長しりとり問題は，各プレイヤが協調して，しりとりが長く続くように単語を出していく言葉遊びととらえることができる．しりとりは人間の発想力を高める題材として有望なものであり，最長しりとりを求めることは今後のアプリケーション開発に寄与すると考えられる．

最長しりとり問題において，国語辞典の見出し語から得られる可能なしりとりを全列挙することは，計算時間の点から不可能である．また，局所探索に代表されるヒューリスティック探索では，最適解が得られる保証はない．これに対して，先行研究[7] では，最長しりとり問題をしりとりに使う単語数を最大化する整数計画問題に帰着し，複数のしりとりによる単語列の単語数最大化を許す緩和問題による分枝限定法を導入することにより，効率的に最適解が得られることが示された．

本論文では，最長しりとり問題を一般化し，しりとりとなる単語列の総文字数を最大とする文字数最大しりとり問題を導入する．最長しりとり問題では，最初の文字および末尾の文字が同じ単語については，それぞれの単語を区別する必要がなかった．一方，文字数最大しりとり問題では，単語の長さによって，それらの単語を区別することが必要となる．文字数最大しり

† 東京農工大学工学部情報コミュニケーション工学科
Department of Computer, Information and Communication Sciences, Faculty of Engineering, Tokyo University of Agriculture and Technology

とり問題では，最長しりとり問題と同じ辞書を使った場合でも，問題を記述するための変数の数が増加し，それによる計算時間の増加が予想される．文字数最大しりとり問題は，最長しりとり問題と同様に，与えられたグラフから最大オイラー路サブグラフを求める問題と等価であり，NP–困難な問題となる[3]が，先行研究において，最長しりとり問題は現実的な時間で最適解を得られた．しかし，文字数最大しりとり問題の場合，現実的な時間で解けるかどうかは未知であり，実験的に検証する必要がある．

文字数最大しりとり問題は最長しりとり問題の自然な拡張であり，両者の最適解を比較することは，日本語の特徴を理解するうえで興味深い．文字数最大しりとり問題は，最長しりとり問題に比べ，一般的な最長路問題に近く，情報工学上の問題として，現実的に解くことができるかどうかという興味もある．

本論文では，最長しりとり問題と文字数最大しりとり問題から派生される問題についても考察する．派生問題としては，文字数を固定した場合の最長しりとり問題と単語数を固定した場合の文字数最大しりとり問題を扱う．これらの問題により，最適解の特徴を分析する．

本論文の構成は次のとおりである．2 章では，本論文で用いるしりとり問題，最長しりとり問題，文字数最大しりとり問題および 2 つの派生問題を定義する．3 章ではネットワーク表現による文字数最大しりとり問題のモデル化について述べる．4 章では，LP ベースの分枝限定法による文字数最大しりとり問題の解法について述べる．5 章では，派生問題の解法について述べる．6 章では，種々の実験結果について述べる．また，最長しりとり問題との比較を行い，7 章では本論文の成果についてまとめる．

## 2. 問題の定義

しりとり問題，最長しりとり問題，本論文で扱う文字数最大しりとり問題，および 2 つの派生問題を次のように定義する．

**しりとり問題**
しりとりは，ある単語から始めて，前の単語の末尾の文字で始まる単語をつないだ単語列である．同じ単語が 2 度以上しりとりに現れてはならない．単語はすべてひらがな表記されているものとする．しりとり問題は，既知の単語集合（辞書）よりしりとりを作成する問題である．

**最長しりとり問題**
最長しりとり問題は，しりとり問題において，単語数が最大となるしりとりを 1 つみつける問題である．

**文字数最大しりとり問題**
文字数最大しりとり問題は，しりとり問題において，単語列の総文字数が最大となるしりとりを 1 つみつける問題である．

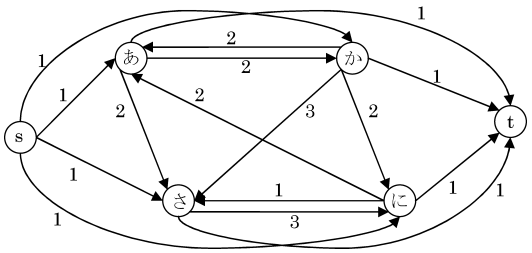
**固定単語長文字数最大しりとり問題**
文字数最大しりとり問題で，単語数が定数により決められている問題である．しりとりとして完結させるため，作成されたしりとりの単語列の前後に，作成されたしりとりに使われなかった辞書中の単語がつながってはならない．

**固定文字数最長しりとり問題**
最長しりとり問題で，しりとりとなる単語列の総文字数が定数により決められている問題である．しりとりとして完結させるため，作成されたしりとりの単語列の前後に，作成されたしりとりに使われなかった辞書中の単語がつながってはならない．

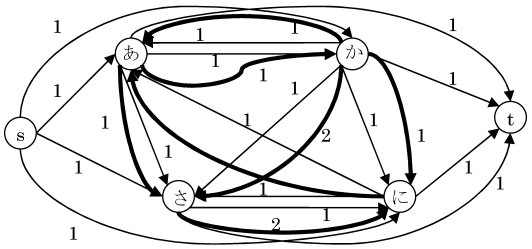
## 3. 文字数最大しりとり問題のモデル化

最長しりとり問題のモデル化は先行研究[7]で行われている．文字数最大しりとり問題では，先行研究におけるモデルを一般化し，単語長ごとにアークを設けたネットワークによりモデル化を行う．

**図 1** に，同じ辞書に対する最長しりとり問題と文字数最大しりとり問題のためのネットワークの例を示す．図 1 a)，b) で，$s$，$t$ を除く頂点は単語の先頭または末尾となるひらがなに対応する．最長しりとり問題のための図 1 a) における頂点 $s$，$t$ を除く頂点間のアークは，始点の文字から終点の文字への単語の存在を示し，アークの重みは辞書に存在する単語数を示す．一方，文字数最大しりとり問題のための図 1 b) における頂点 $s$，$t$ を除く頂点間のアークは，始点の文字から終点の文字へのある文字列長の単語の存在を示し，アークの重みは辞書に存在するその文字列長の単語数を示す．つまり，b) においては，頂点間には単語の長さの種類数分の並列アークが存在する．たとえば，図 1 a) では頂点「あ」から頂点「か」へは重み 2 のアークが 1 本だけ存在するが，図 1 b) では，単語長 1 で重み 1 のアークが 1 本と，単語長 2 で重み 1 のアークが 1 本




a) 最長しりとり問題におけるネットワーク表現
各アークの数字は辞書中の単語数を表す




b) 文字数最大しりとり問題におけるネットワーク表現
各アークの数字は辞書中の単語数を表す
細線は 1 文字の単語，太線は 2 文字の単語のアークを表す

**図 1** 最長しりとり問題，文字数最大しりとり問題のネットワーク表現

Fig. 1 Network models of the longest shiritori problem and the maximum-shiritori-string-length shiritori problem.

の 2 本の並列アークが存在し，単語長によりアークが区別されている．

図 1 a)，b) いずれにおいても，文字に対応する頂点およびそれらの頂点間のアーク以外に，頂点 $s$ と，頂点 $s$ からすべての文字に対応する頂点への重み 1 のアーク，および，頂点 $t$ と，すべての文字に対応する頂点から頂点 $t$ への重み 1 のアークが加えられている．これらの頂点とアークは，任意の 2 頂点間における問題を解くための追加であり，詳細は 4 章で述べる．

図 1 a) において，最長しりとりは，アークの重みを各アークの容量ととらえ，頂点 $s$ から頂点 $t$ へのフローで，アークを流れるフローの総和を最大化して得られるフローの流れたアーク集合から構成できた．このとき，フローが流れたアークによる部分グラフが連結グラフであれば，最長しりとり問題の許容解となる．一方，図 1 b) において，文字数最大しりとりは，アークの重みを各アークの容量ととらえ，頂点 $s$ から頂点 $t$ へのフローで，アークを流れるフローに，そのアークの種類に対応する文字数（フローの重み）を乗じた値の総和を最大化して得られるフローの流れたアーク集合から構成できる．このとき，フローの流れたアークによる部分グラフが連結グラフであれば，文字数最大しりとり問題の許容解となる．

次に，文字数最大しりとり問題の定式化について述べる．文字数最大しりとり問題を記述するために，文字，単語の長さおよび単語数を扱う．単語の初めの文字または終わりの文字になる文字集合を $V$ とおく．ここでは，文字を数字に対応付けて表現し，文字数 $n$ の集合を次のように表す．

$$V = \{1, 2, \ldots, n\}$$

単語の長さの集合を $L$ とおく．本論文では既知の単語を扱うため，単語の長さは有限であり，最大の単語の長さが $l$ の場合，次のように表す．

$$L = \{1, 2, \ldots, l\}$$

$V \times V \times L$ によって表される集合は文字数最大しりとり問題を解くための単語情報を表す．あるひらがな $i \in V$ を始点，$j \in V$ を終点とし，長さが $k \in L$ であるアークを $a_{ij}^k$，このアークに対する単語の数を $f_{ij}^k$ と記述する．アークの集合を $A$，単語数の集合を $F$ で表す．これによって，ネットワーク $N = (G = (V \cup \{s,t\}, A), F)$ が得られる．ただし，$f_{si}^1 = f_{it}^1 = 1$，$f_{si}^k = f_{it}^k = 0 (k > 1)$，$i \in V$，$k \in L$ である．このとき，文字数最大しりとり問題は次のように定式化される．

**文字数最大しりとり問題のネットワークによる定式化**

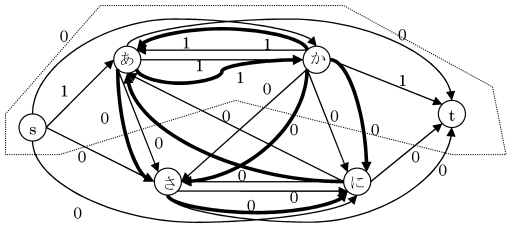
$x_{ij}^k$ をアーク $a_{ij}^k$ におけるフロー量とする．有向ネットワーク $N = (G = (V \cup \{s,t\}, A), F)$ において，頂点 $s$ からフロー量 1 のフローが流出し，頂点 $t$ へフロー量 1 のフローが流入する．このとき，フローが流れているアーク $(x_{ij}^k > 0)$ により誘導される部分グラフ（以後，**フロー誘導部分グラフ**と呼ぶ）は連結とする条件のもとで，$\sum_{i \in V \cup \{s\}} \sum_{j \in V \cup \{t\}} \sum_{k \in L} k x_{ij}^k, 0 \le x_{ij}^k \le f_{ij}^k$ を最大とするフローを求める問題．

ここでフローは次のように特徴付けられる．

$$\sum_{j \in V \cup \{s,t\}} \sum_{k \in L} x_{ij}^k - \sum_{j \in V \cup \{s,t\}} \sum_{k \in L} x_{ji}^k = \begin{cases} 1 & : i = s \\ 0 & : i \in V \\ -1 & : i = t \end{cases}$$

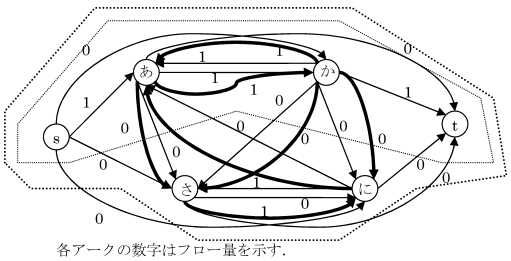
$$0 \le x_{ij}^k \le f_{ij}^k \ \forall i, j \in V \cup \{s,t\}, \forall k \in L$$

$f_{ij}^k = 0$ であるアークはしりとりで使うことができない．本論文では表現を簡単にするためにこの条件を持つアークのことを「アークがない」と表現する．




各アークの数字はフロー量を示す．
a) 点線内の頂点へ接続するアークのフロー量の総和が 1 以上
b) 点線外の頂点へのアークのフロー量の総和が 0 である
である点線で囲まれたような s, t を含む頂点集合が一意に決定される場合，フロー誘導部分グラフは連結である

**図 2** 部分ネットワークによる分割
Fig. 2 Division by sub-network.




各アークの数字はフロー量を示す．
点線で囲まれた頂点集合のように，図 2 とは異なり，2 通りの頂点集合が存在するので，フロー誘導部分グラフは連結ではない

**図 3** 連結でないフローの例
Fig. 3 An example of disconnected network.

## 4. 文字数最大しりとり問題の解法

本章では，文字数最大しりとり問題を解くための分枝限定法について述べる．ネットワークによる定式化において，しりとりが構成されるフローの条件は前章で述べたように以下で示される．

(1) 頂点 $s$ からフロー量 1 のフローが流出し，頂点 $t$ へフロー量 1 のフローが流入する．
(2) フロー誘導部分グラフが連結である．

条件 (1) は頂点数の制約式で記述することができるが，条件 (2) には頂点の数に対し指数オーダの制約式が必要となるため，現実的に記述することはできない．これは，可能なフロー誘導部分グラフのとりかたを制約条件として列挙しなければならないためである．頂点 $s$ と頂点 $t$ を含む頂点集合 $U$ をとったとき，次の条件を満たす $U$ が一意に決定される場合，ネットワーク上のフロー誘導部分グラフは連結であるといえるが，一意に決定されない場合は非連結である．

$$\sum_{j\in U-\{s\}}\sum_{k\in L} x_{ij}^k > 0 \quad \forall i \in U-\{t\}$$

$$\sum_{k\in L} x_{ij}^k = 0 \quad \forall i \in U-\{t\}, \forall j \in V-U$$

**図 2** はフロー誘導部分グラフが連結な場合の例であり，**図 3** は非連結な場合の例である．図 3 では,「さ」と「に」を結ぶアークにフローがあるので，2 通りの方法で $U$ を決めることができる．このような条件を整数計画問題の枠組みで記述すると，頂点集合のとりかたは頂点数の指数オーダで増加するため，制約条件および選ばれる頂点集合の一意性を記述するための変数も頂点数の指数オーダで増加する．

このような直接的な問題記述をさけ，本論文では文字数最大しりとり問題のための緩和問題を繰り返し解く分枝限定法を用いる．先のフローの条件 (1) は容易に記述できることを利用する．フローの条件 (1) で文字数を最大とする問題を解く場合，(2) の条件は考慮されないので，フロー誘導部分グラフが連結でない場合がある．このような場合，後述するように，(1) の条件だけで得られたフロー誘導部分グラフの非連結な部分をつなげるような条件を付加することで，非連結な部分をつなぐ解の構成を試みる．このような制約が繰り返し追加される際，解かれる緩和問題 **(RP$_e$)**，$e = 0, 1, \ldots$ を次に示す．$V_m$ は緩和問題 **(RP$_m$)** の解によるフロー誘導部分グラフで，$s$ から $t$ をつなぐ経路上にある頂点の集合を表す．

**(RP$_e$)** 最大化 $z = \displaystyle\sum_{i\in V\cup\{s\}}\sum_{j\in V\cup\{t\}}\sum_{k\in L} k x_{ij}^k$

条件
$$\sum_{j\in V} x_{sj}^1 = 1$$
$$\sum_{j\in V}\sum_{k\in L} x_{ij}^k - \sum_{j\in V}\sum_{k\in L} x_{ji}^k = 0 \quad \forall i \in V$$
$$\sum_{j\in V} x_{jt}^1 = 1$$
$$\sum_{i\in V_m}\sum_{j\in V-V_m}\sum_{k\in L} x_{ij}^k \geq 0 \qquad \forall m = 0, 1, \ldots, e-1$$
$$0 \leq x_{ij}^k \leq f_{ij}^k, \quad \forall i \in V, \forall j \in V, \forall k \in L$$
$$0 \leq x_{sj}^1 \leq 1, \quad \forall j \in V$$
$$0 \leq x_{jt}^1 \leq 1, \quad \forall j \in V$$
$$x_{ij}^k \in \mathbf{Z}, \quad \forall i \in V, \forall j \in V, \forall k \in L$$
$$x_{sj}^1 \in \mathbf{Z}, \quad \forall j \in V$$
$$x_{jt}^1 \in \mathbf{Z}, \quad \forall j \in V$$

最初の緩和問題 **(RP$_0$)** は，フローに関する条件 (1) のもとで文字数を最大にする問題となっている．この問題の目的関数は各アークのフロー量とそのアークの単語長をかけたものの和になり，全体の文字数を最大とすることを表している☆．各アークのフロー量

☆ 単にフロー量の和であれば，最長しりとり問題を解く．

は辞書でカウントされた単語の数を上限として，0 以上の条件が設定される．制約条件は，$x_{ij}^k$ がフローとなる条件そのものである．

緩和問題（$\mathbf{RP}_e$）によって連結なフロー誘導部分グラフが得られない場合は，分枝操作によって，解を得る．緩和問題（$\mathbf{RP}_e$）の解によるフロー誘導部分グラフにおいて，$s$ から $t$ への経路上の頂点集合を $V_e$ とおく．このとき，この補集合 $V - V_e$ の要素である頂点にフロー誘導部分グラフのアークが存在する場合，連結ではない．このとき，問題は次のように分割（分枝操作）できる．

(1) 頂点集合 $V_e$ から $V - V_e$ へのフローが存在しない文字数最大しりとり問題
(2) 頂点集合 $V_e$ から $V - V_e$ へのフローが存在する文字数最大しりとり問題

(1) の場合，$V_e$ に含まれる頂点だけを用いた文字数最大しりとり問題の最適解は緩和問題（$\mathbf{RP}_e$）の解として得られている．よって，(2) だけを繰り返し解けばよい．(2) により，緩和問題（$\mathbf{RP}_{e+1}$）は，$V_e$ から $V - V_e$ へのアークが少なくとも 1 本存在するという条件を（$\mathbf{RP}_e$）に追加することで得られる．

フローに関する条件だけで構成される問題（$\mathbf{RP}_0$）は，制約条件の係数が 1，0 のいずれかであり，その任意の正方小行列の行列式の値が 1，0，$-1$ となるため，Totally Unimodular Matrix（**TU**）[6] となる．制約条件の係数行列が **TU** である線形計画問題では整数基底解が得られることが知られている．そのため，問題（$\mathbf{RP}_0$）は整数条件を考慮せずに解くことができる．これに対し，問題（$\mathbf{RP}_e$），$e > 0$ では，付け加えられた制約条件のため，制約条件の係数行列が **TU** にならない．そのため，整数解が得られない場合は，整数計画問題として整数解を求めることが必要となる．

以上，説明した緩和問題（$\mathbf{RP}_e$），$e >= 0$ を用いて，分枝操作により，文字数最大しりとり問題を解くアルゴリズムを以下に示す．このアルゴリズムのことを LP ベースの分枝限定法と呼ぶ．ここで，**MS3** は，Maximum-Shiritori-String-Length Shiritori の略で，文字数最大しりとりを意味する．

**Algorithm** Making-MS3
**begin**
  {$\boldsymbol{x}^*$:最適解を保持 }
  $e := 0$; { 分枝の回数 }
  $z^* := 0$; { $z^*$:暫定値 }
  **while** true **do**
  **begin**
    （$\mathbf{RP}_e$）を線形計画問題として解く;
    **if** （$\mathbf{RP}_e$） の解が整数解でない **then**
      （$\mathbf{RP}_e$）を整数計画問題として解く;
    **if** （$\mathbf{RP}_e$） に実行可能解がない **then**
      **goto 1**;
    $\boldsymbol{x}$ :=（$\mathbf{RP}_e$）の解;
    $y$ :=（$\mathbf{RP}_e$）の解の $s$,$t$ をつなぐ文字数;
    $z$ :=（$\mathbf{RP}_e$）の解の目的関数値;
    **if** $z = y$ **then** { フロー誘導部分グラフが連結であった場合 }
      **begin**
        **if** $z^* < z$ **then**
        **begin**
          $\boldsymbol{x}^* := \boldsymbol{x}$;
          $z^* := z$;
        **end**;
        **goto 1**;
      **end**
    **else** { フロー誘導部分グラフが連結でない場合 }
      **begin**
        **if** $z < z^*$ **then**
          **goto 1**;
        **if** $z^* < y$ **then**
        **begin**
          $\boldsymbol{x}^* := \boldsymbol{x}$ より，$s,t$ 間の連結成分を抽出;
          $z^* := y$;
        **end**;
        $e := e + 1$;
      **end**
  **end**
**1:** $\boldsymbol{x}^*$ よりしりとりを構成する
  { 文字数最大しりとりの文字数は，$z^* - 2$ }
**end**;

アルゴリズム Making-MS3 において，変数 $\boldsymbol{x}$ は問題（$\mathbf{RP}_e$）の解を保持し，$\boldsymbol{x}^*$ は暫定解を保持する．変数 $z$ は問題（$\mathbf{RP}_e$）の目的関数の値を保持し，$z^*$ は暫定値を保持する．変数 $y$ は問題（$\mathbf{RP}_e$）の解 $\boldsymbol{x}$ における $s$ と $t$ を結ぶしりとり上での文字数を保持する．このしりとりの通る頂点が $V_e$ となる．解 $\boldsymbol{x}^*$ からしりとりを構成する方法は，先行研究[7] と同じである．

## 5. 派生問題

本章では，2 章で定義した 2 つの派生問題，固定単語長文字数最大しりとり問題，固定文字数最長しりとり問題の解法について述べる．これらは，前章で述べた緩和問題に，制約条件を付加することで実現できる．最長しりとり問題および文字数最大しりとり問題では，最長および文字数最大の最適性により，作成されたしりとりで使われなかった単語を使ってより長い，あるいは文字数の多いしりとりは構成できない．これに対

し，派生問題の場合，単純に単語数や文字数による制約を条件として付加しただけでは，作成されたしりとりの前後に使われていない単語がつながるようなしりとりが構成できる．そこで，作成されたしりとりの単語列の前後につながる単語が，作成されたしりとりに使われなかった辞書中の単語集合に存在しないことを条件として記述する必要がある．

先頭および末尾に関するこの条件は，先頭となる文字，つまり，$s$ からのアークの終点となる文字で終わる単語はすべて使われているとともに，末尾となる文字，つまり，$t$ へのアークの始点となる文字から始まる単語もすべて使われていることを表すので，次のように記述できる（**しりとりを完結するための条件**）．

$$\sum_{j\in V}\sum_{k\in L} x_{ji}^k - x_{si}^1 \sum_{j\in V}\sum_{k\in L} f_{ji}^k \geq 0,\ \forall i \in V$$

$$\sum_{j\in V}\sum_{k\in L} x_{ij}^k - x_{it}^1 \sum_{j\in V}\sum_{k\in L} f_{ij}^k \geq 0,\ \forall i \in V$$

ここで，$x_{si}^1$ は 0 または 1 の値をとる整数変数であり，1 の場合，$i$ がしりとりの先頭となるひらがなを表す．そして，$x_{it}^1$ は 0 または 1 の値をとる整数変数であり，1 の場合，$i$ がしりとりの末尾となるひらがなを表す．

固定単語長文字数最大しりとり問題の場合，しりとりを完結するための制約条件に加えて，緩和問題（$\mathbf{RP}_e$）に次の条件を付加することになる．

$$\sum_{i\in V\cup\{s\}}\sum_{j\in V\cup\{t\}}\sum_{k\in L} x_{ij}^k = \text{固定単語長}$$

3 章で述べたように，文字数最大しりとり問題の解法における目的関数において，フローに対する重みをすべて 1 にした場合，最長しりとり問題を解くことになる．よって，固定文字数最長しりとり問題については，次のように緩和問題（$\mathbf{RP}_e$）を変形することで，緩和問題を得る．

目的関数

$$z = \sum_{i\in V\cup\{s\}}\sum_{j\in V\cup\{t\}}\sum_{k\in L} x_{ij}^k$$

しりとりを完結するための条件に加えて追加される制約条件

$$\sum_{i\in V\cup\{s\}}\sum_{j\in V\cup\{t\}}\sum_{k\in L} k x_{ij}^k = \text{固定文字数}$$

また，前章で述べた LP ベースの分枝限定法の変数 $y$ は，（$\mathbf{RP}_e$）の解における $s$，$t$ をつなぐしりとりの文字数であったが，固定文字数最長しりとり問題を解くために，（$\mathbf{RP}_e$）の解における $s$，$t$ をつなぐしりとりの単語数となる．

LP ベースの分枝限定法に関しては，フロー誘導部分グラフが連結でない場合，目的関数の値と $s$，$t$ を結ぶ連結なフロー誘導部分グラフの値が異なる．この場合，$s$，$t$ を結ぶ連結なフロー誘導部分グラフは，しりとり全体の文字数，または単語数に関する条件を満たしていないことになる．したがって，次の 2 つの条件によって，分枝操作を行う．

(1) 頂点集合 $V_e$ から $V - V_e$ へのフローが存在しない派生問題

(2) 頂点集合 $V_e$ から $V - V_e$ へのフローが存在する派生問題

(2) は，文字数最大しりとり問題の場合と同様であるが，(1) の場合は，頂点を $V_e$ 内に限定して，制約条件を満たす解を探すことになる．このため，頂点を $V_e$ の要素に限定して，緩和問題を解き直す．アルゴリズム Making-MS3 におけるフロー誘導部分グラフが連結でなかった場合の処理で，$s$ と $t$ を結ぶフロー誘導部分グラフの単語数あるいは文字数である $y$ を使った暫定解の更新判定が行えない．これは，$s$ と $t$ を結ぶ連結なフロー誘導部分グラフでなければ，許容解とならないためである．これらの要因と緩和問題に追加された条件のため，文字数最大しりとり問題より，派生問題は解きにくいことが予想される．

**表 1** 広辞苑と ICOT 辞書のデータ
Table 1 Information about Koujien and ICOT dictionary used in experiment.

| | 広辞苑 | ICOT 辞書（名詞） |
|---|---|---|
| 単語数（W） | 192,687 | 110,922 |
| 文字種類数 | 70 | 70 |
| 始めの文字種類数（S） | 70 | 69 |
| 終りの文字種類数（T） | 70 | 70 |
| アーク数（A） | 4205 | 3708 |
| $\frac{A}{ST}$ | 86% | 77% |
| アーク数に対する最長しりとり長（L） | 3907 | 3280 |
| $\frac{L}{A}$ | 92% | 88% |
| 最長しりとり長（Q） | 86788 | 51351 |
| $\frac{Q}{W}$ | 45% | 46% |

## 6. 実　　験

前章で述べた手法を評価するための実験を行った．実装は，Windows XP 上の Cygwin（ver. 2.218）で gcc（ver. 3.2）を用い，整数計画法のソルバとして，GNU GLPK（ver. 4.2）を使った．

本章で述べる実験では，広辞苑[4] から抜き出された名詞（広辞苑）と ICOT 形態素解析辞書[5]（ICOT 辞書）中の全単語を用いた．それぞれの辞書のデータを**表 1** に示す．

### 6.1 単語数の変化に対する評価実験

辞書中の単語数が変化した際の解法の性能の評価実

**表 2** 各アークの単語数を半減させた場合の結果（広辞苑）
Table 2 Results of using the half number of words recursively (Koujien).

| 頂点数 | | | | アーク数 | 単語数 | 文字数最大しりとり問題 | | | | 最長しりとり問題 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All | InOut | In | Out | | | Relax | Time | Len | Char | Relax | Time | Len | Char |
| 68 | 68 | 68 | 68 | 23,070 | 192,687 | 1 | 55.672 | 83900 | 436106 | 1 | 51.797 | 86796 | 406971 |
| 68 | 66 | 67 | 67 | 15,776 | 89,345 | 1 | 20.109 | 36052 | 181824 | 1 | 19.86 | 37285 | 170750 |
| 68 | 66 | 67 | 67 | 9,806 | 39,709 | 2 | 5.892 | 14107 | 69432 | 2 | 5.438 | 14554 | 65612 |
| 68 | 64 | 65 | 67 | 5,415 | 16,600 | 2 | 1.218 | 4808 | 23139 | 3 | 1.218 | 4949 | 21842 |
| 64 | 54 | 56 | 62 | 2,636 | 6,416 | 15 | 0.641 | 1339 | 6402 | 6 | 0.36 | 1378 | 6124 |
| 61 | 33 | 34 | 60 | 1,136 | 2,255 | 54 | 7.531 | 310 | 1463 | 60 | 4.313 | 314 | 1368 |
| 51 | 14 | 15 | 50 | 433 | 714 | 1 | 0.047 | 48 | 217 | 1 | 0.031 | 48 | 199 |
| 34 | 4 | 8 | 30 | 132 | 184 | 1 | 0.031 | 6 | 27 | 1 | 0.032 | 6 | 25 |
| 16 | 0 | 4 | 12 | 32 | 38 | 1 | 0.016 | 1 | 6 | 1 | 0.03 | 1 | 4 |
| 6 | 0 | 2 | 4 | 6 | 6 | 1 | 0.031 | 1 | 5 | 1 | 0.031 | 1 | 4 |

**表 3** 各アークの単語数を半減させた場合の結果（ICOT 辞書）
Table 3 Results of using the half number of words recursively (ICOT).

| 頂点数 | | | | アーク数 | 単語数 | 文字数最大しりとり問題 | | | | 最長しりとり問題 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All | InOut | In | Out | | | Relax | Time | Len | Char | Relax | Time | Len | Char |
| 68 | 67 | 68 | 67 | 15,184 | 110,922 | 1 | 18.468 | 49041 | 231525 | 1 | 16.922 | 51364 | 214864 |
| 68 | 67 | 68 | 67 | 9,893 | 50,760 | 1 | 5.828 | 21030 | 94496 | 1 | 5.875 | 21862 | 87987 |
| 68 | 63 | 65 | 66 | 5,870 | 22,170 | 1 | 1.5 | 8279 | 35815 | 1 | 1.39 | 8512 | 33335 |
| 66 | 55 | 57 | 64 | 3,186 | 9,116 | 1 | 0.406 | 2895 | 12302 | 1 | 0.376 | 2960 | 11488 |
| 62 | 42 | 45 | 59 | 1,644 | 3,470 | 1 | 0.141 | 901 | 3782 | 1 | 0.125 | 919 | 3567 |
| 58 | 25 | 28 | 55 | 715 | 1,146 | 1 | 0.063 | 201 | 835 | 1 | 0.063 | 202 | 787 |
| 42 | 12 | 16 | 38 | 221 | 294 | 1 | 0.047 | 22 | 87 | 1 | 0.03 | 22 | 84 |
| 22 | 4 | 8 | 18 | 48 | 56 | 1 | 0.03 | 4 | 17 | 1 | 0.016 | 4 | 16 |
| 6 | 0 | 2 | 4 | 8 | 8 | 1 | 0.031 | 1 | 5 | 1 | 0.031 | 1 | 4 |

験を行った．まず，広辞苑と ICOT 辞書の全単語について，文字数最大しりとり問題を解いた．その後，それぞれの辞書に対して次の実験を再帰的に繰り返した．

(1) 各アークの単語数を半分にしたデータによる実験
(2) アークをランダムに削除することで全体のアーク数を半分にしたデータによる実験

#### 6.1.1 アークの単語数を半減した実験

本項では，各アークに付加された単語数を半分にしていった場合の結果について示す．単語数を半分にする際に，小数点以下は切捨てにしているので，半分にすることで全体の単語数は元の単語数の半分以下になっている．**表 2**，**表 3** に結果を示す．本実験では，最長しりとり問題も文字数最大しりとり問題と同じモデルで解いている．表の各項目は次の内容を表している．

- **All**：アークが存在する頂点数
- **InOut**：出るアークと入るアークが存在する頂点数
- **In**：入るアークが存在する頂点数
- **Out**：出るアークが存在する頂点数
- **アーク数：**データ中の単語数が 1 個以上のアーク数
- **単語数：**データ中の総単語数
- **Relax**：最適解を得るために生成された緩和問題数
- **Time**：最適解を得るためにかかった時間（秒）
- **Len**：作成されたしりとりに含まれる単語数
- **Char**：作成されたしりとりに含まれる文字数

広辞苑に関する表 2 の結果では，特定のデータについて緩和問題の数が増えていた．先行研究[7] では，緩和問題の数は多くても 4 個程度で最適解が得られていた．本実験の場合はそれを上回る緩和問題が生成されている．緩和問題が $m$ 個できるということは，ネットワークを $m$ 回空でない 2 つの部分ネットワークに分割されたことを表しており，スパースなネットワークとなっていることを意味する．しかしながら，表 3 の場合はそのような解が得られていない．これは，辞書の違いであると考えられる．緩和問題を生成していく際，LP ベースの分枝限定法において，暫定解を保持する変数 $\boldsymbol{x}^*$ の更新は行われていなかった．これは，初期の緩和問題ですでに最適解が得られていることを示している．

すべての場合で，同じ条件ならば，当然，文字数最大

**表 4** 各アーク数をランダムに半減させた場合の結果（広辞苑）
Table 4 Results of using the half number of arcs recursively (Koujien).

| 頂点数 | | | | アーク数 | 単語数 | 文字数最大しりとり問題 | | | | 最長しりとり問題 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All | InOut | In | Out | | | Relax | Time | Len | Char | Relax | Time | Len | Char |
| 68 | 68 | 68 | 68 | 23,070 | 192,687 | 1 | 55.281 | 83900 | 436106 | 1 | 51.97 | 86796 | 406971 |
| 68 | 67 | 68 | 67 | 11,535 | 98,538 | 1 | 10.641 | 41821 | 216831 | 1 | 9.687 | 43236 | 202166 |
| 68 | 67 | 68 | 67 | 5,768 | 49,852 | 1 | 2.048 | 20395 | 107452 | 1 | 1.906 | 21095 | 100117 |
| 68 | 67 | 68 | 67 | 2,884 | 24,687 | 1 | 0.531 | 10008 | 53064 | 1 | 0.485 | 10349 | 49640 |
| 68 | 66 | 67 | 67 | 1,442 | 11,902 | 1 | 0.172 | 4546 | 24310 | 1 | 0.172 | 4690 | 22526 |
| 68 | 64 | 65 | 67 | 721 | 5,766 | 1 | 0.078 | 1912 | 10100 | 1 | 0.078 | 1984 | 9620 |
| 68 | 60 | 64 | 64 | 361 | 3,008 | 1 | 0.047 | 599 | 3193 | 1 | 0.047 | 627 | 3005 |
| 65 | 55 | 59 | 61 | 181 | 1,865 | 10 | 0.297 | 173 | 909 | 4 | 0.11 | 177 | 887 |
| 61 | 36 | 48 | 49 | 91 | 424 | 31 | 1.282 | 29 | 184 | 217 | 46.407 | 29 | 181 |
| 49 | 17 | 32 | 34 | 46 | 177 | 51 | 1.233 | 6 | 52 | 51 | 1.234 | 6 | 49 |
| 34 | 8 | 21 | 21 | 23 | 101 | 1 | 0.031 | 2 | 17 | 1 | 0.031 | 3 | 12 |
| 20 | 2 | 10 | 12 | 12 | 58 | 1 | 0.032 | 1 | 13 | 1 | 0.032 | 2 | 11 |
| 11 | 0 | 5 | 6 | 6 | 38 | 1 | 0.016 | 1 | 9 | 1 | 0.031 | 1 | 3 |
| 6 | 0 | 3 | 3 | 3 | 24 | 1 | 0.016 | 1 | 5 | 1 | 0.031 | 1 | 5 |
| 4 | 0 | 2 | 2 | 2 | 23 | 1 | 0.016 | 1 | 5 | 1 | 0.031 | 1 | 5 |

**表 5** 各アーク数をランダムに半減させた場合の結果（ICOT 辞書）
Table 5 Results of using the half number of arcs recursively (ICOT).

| 頂点数 | | | | アーク数 | 単語数 | 文字数最大しりとり問題 | | | | 最長しりとり問題 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| All | InOut | In | Out | | | Relax | Time | Len | Char | Relax | Time | Len | Char |
| 68 | 67 | 68 | 67 | 15,184 | 110,922 | 1 | 18.548 | 49041 | 231525 | 1 | 16.641 | 51364 | 214864 |
| 68 | 67 | 68 | 67 | 7,592 | 55,511 | 1 | 3.843 | 24097 | 113771 | 1 | 3.422 | 25231 | 105748 |
| 68 | 67 | 68 | 67 | 3,796 | 27,359 | 1 | 0.828 | 11636 | 54720 | 1 | 0.735 | 12169 | 50281 |
| 68 | 67 | 68 | 67 | 1,898 | 14,605 | 1 | 0.234 | 6071 | 28634 | 1 | 0.25 | 6323 | 27097 |
| 68 | 65 | 66 | 67 | 949 | 7,354 | 1 | 0.11 | 2666 | 13157 | 1 | 0.094 | 2786 | 12463 |
| 67 | 63 | 65 | 65 | 475 | 3,908 | 1 | 0.047 | 986 | 5012 | 1 | 0.062 | 1043 | 4772 |
| 65 | 55 | 60 | 60 | 238 | 1,639 | 1 | 0.032 | 314 | 1749 | 1 | 0.047 | 325 | 1663 |
| 61 | 43 | 50 | 54 | 119 | 852 | 27 | 1.077 | 57 | 305 | 2 | 0.047 | 63 | 302 |
| 49 | 26 | 35 | 40 | 60 | 464 | 28 | 0.751 | 17 | 119 | 31 | 0.765 | 17 | 119 |
| 39 | 8 | 22 | 25 | 30 | 358 | 1 | 0.031 | 2 | 29 | 1 | 0.032 | 4 | 13 |
| 27 | 0 | 13 | 14 | 15 | 184 | 1 | 0.032 | 1 | 8 | 1 | 0.031 | 1 | 7 |
| 16 | 0 | 8 | 8 | 8 | 140 | 1 | 0.016 | 1 | 8 | 1 | 0.031 | 1 | 7 |
| 8 | 0 | 4 | 4 | 4 | 135 | 1 | 0.015 | 1 | 8 | 1 | 0.031 | 1 | 7 |
| 4 | 0 | 2 | 2 | 2 | 9 | 1 | 0.016 | 1 | 8 | 1 | 0.031 | 1 | 8 |

しりとり問題の解の方が文字数が多く，最長しりとり問題の方が単語数が多くなっている．広辞苑で，文字数最大しりとり問題を解いた場合の単語数は 83,900，最長しりとり問題を解いた場合の単語数は 86,796 であり，その差は 2,896 単語となった．

最長しりとり問題の計算時間は，先行研究と比べ，10 倍以上となっている．特に，広辞苑に関しては約 41 倍の計算時間がかかる．これは，最長しりとり問題を解く際，最大文字数しりとり問題と同じモデルを用いたため，モデルを記述するための変数の数が最大の単語長の倍数となっていることが原因と考えられる．

### 6.1.2 アーク数を半減した実験

本項では，前項のように単語数ではなく，アーク数を半減した実験について述べる．これは，単語が 1 以上あるアークの中から半分をランダムに選び，単語の頻度を 0 にすることで実現している．削減するアーク数は小数点以下切り捨てで半減させる．この実験の結果を**表 4** と**表 5** に示す．項目の意味は前項で説明したとおりである．

ICOT 辞書のデータでアークを半分にしていった表 5 では，ICOT 辞書のデータで各アークの単語数を半分にしていった表 3 よりも，緩和問題の数は増えている場合がある．これは，表 3 ではどのアークの単語数も一様に半減するのに対し，表 5 では，アークがいきなりなくなるので，よりスパースなネットワークになりやすいことが原因と考えられる．

広辞苑についての実験結果である表 2 と表 4 より，文字数最大しりとり問題について，しりとりに含まれる単語の 1 単語あたりの平均文字数を調べると，単語数を半分にした場合は，徐々に平均文字数が減少していくのに対し，ランダムにアークを削除した場合は，文字数が増えていく傾向にあった．最長しりとり問題

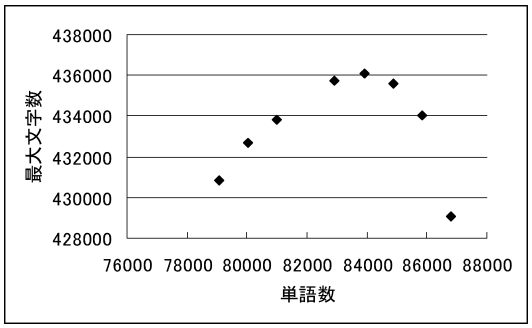



**図 4** 固定単語長文字数最大しりとり問題（広辞苑）
Fig. 4 Fixed-length maximum-shiritori-string-length shiritori problem (Koujien).

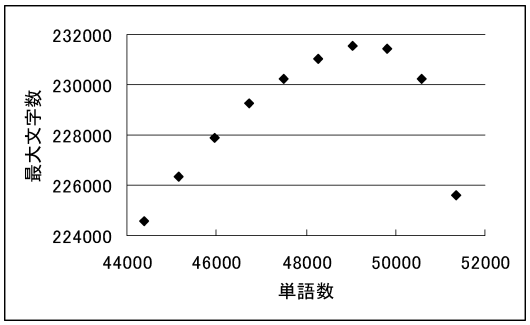



**図 5** 固定単語長文字数最大しりとり問題（ICOT 辞書）
Fig. 5 Fixed-length maximum-shiritori-string-length shiritori problem (ICOT).

で得られた単語数に対する文字数最大しりとり問題の解を与えるしりとりの長さの割合を見ると，単語数を半分にした場合は，割合が増加するのに対し，ランダムにアークを削除した場合は，割合がほぼ一定であった．これは ICOT 辞書でも同様である．これらの実験結果は，日本語の単語では文字数の多い単語を表すアークが相対的に多くないため乱数ではそれらのアークが消去されにくいが，文字数最大しりとり問題で文字数を多くするのに重要な役割を果しているためと考えられる．

結果的に，いずれの実験においても，現実的な時間で解を求められることから，本論文で用いた緩和問題による解法はうまく動作するといえる．

### 6.2 最長しりとり問題と文字数最大しりとり問題に対する最適解の分析

最長しりとり問題の最適解と文字数最大しりとり問題の最適解との関係を調べるために，派生問題を用いて，広辞苑と ICOT 辞書の全単語について，次のことを調べた．固定単語長文字数最大しりとり問題については，文字数が最大となる前後の単語数に関する最

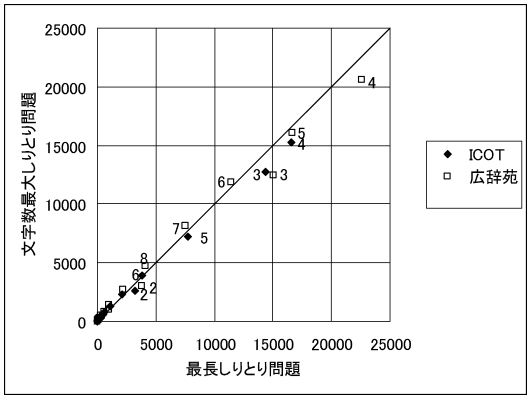



**図 6** 最長しりとりと文字数最大しりとりに関する単語長ごとの単語頻度の比較（グラフ中の数字は単語長）
Fig. 6 Comparison of frequencies of words by length between a maximum-shiritori-string-length shiritori with the longest shiritori length and a maximum-shiritori-string-length shiritori.

大文字数を調べた．固定文字数最長しりとり問題については，単語数が最大となる文字数の前後についての最大単語数を調べた．

固定単語長文字数最大しりとり問題では，表 2 および表 3 より文字数が最大となったときの単語長周辺で，固定単語長を変動させた．広辞苑を使った場合の結果を**図 4**，ICOT 辞書を使った場合の結果を**図 5** に示す．これらのグラフで，最も右に位置するプロットは，最長しりとり問題で得られた最長しりとり長におけるしりとりの最大文字数を表す．そして，右から 4 番目のプロットは，文字数最大しりとり問題の解としてのしりとりの単語数とその文字数の関係を示している．

図 4 および図 5 から，グラフは文字数最大しりとり問題の解となる単語数をピークとした山形をしていることが分かる．頂上の右側は，左側に比べ，傾斜がきつい．これは，単語数の多いしりとりを作るためには，文字数を少なくする単語を使う必要があることを示している．

**図 6** に，最長しりとり，および文字数最大しりとりを構成する単語の頻度を単語長ごとにプロットした．このグラフでは，広辞苑，ICOT 辞書に含まれる単語すべてを用いた．グラフ中の数字は単語長を表し，たとえば，右上のプロットは広辞苑の長さ 4 の単語について最長しりとりでの頻度が文字数最大しりとりでの頻度より多いことを示す．最長しりとりについては，使われる単語の長さが決定されない．文字数最大しりとりと比較するために，長い単語長の単語から順に用いるとして，頻度を算出した．基本的に頻度の少

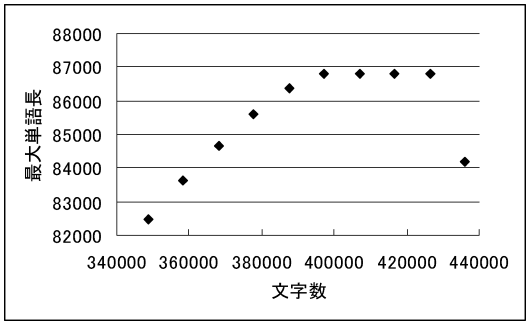



**図 7** 固定文字数最長しりとり問題（広辞苑）
Fig. 7 Fixed-shiritori-string-length longest-shiritori problem (Koujien).

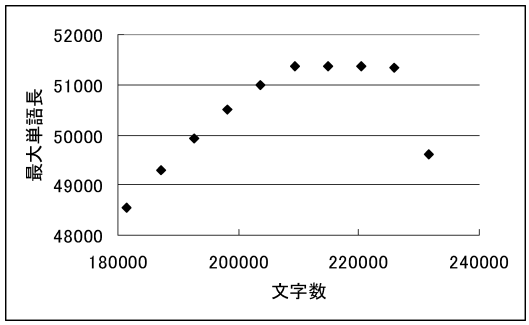



**図 8** 固定文字数最長しりとり問題（ICOT）
Fig. 8 Fixed-shiritori-string-length longest-shiritori problem (ICOT).

ない単語は単語長の長い単語であり，頻度の多い単語は単語長の短い単語を示している．図 6 から，長いしりとりを作るためには短い単語が多く使われ，文字数の多いしりとりを作るためには長い単語が多く使われることが分かる．広辞苑，ICOT 辞書ともに，単語長が 5 以下の場合は最長しりとりにおける頻度が多く，6 以上の場合は文字数最大しりとりにおける頻度が多くなっていた．また，相対的な頻度を単語長ごとに $\frac{\text{文字数最大しりとりでの頻度}}{\text{最長しりとりでの頻度}}$ によって表すと，単語長が短くなるにつれ，この値は小さくなり，長くなるにつれ，大きくなることが分かった．

固定文字数最長しりとり問題では，表 2 および表 3 より，単語長が最大となる文字数の周辺で，固定文字数を変動させた．広辞苑を使った場合の結果を**図 7**，ICOT 辞書を使った場合の結果を**図 8** に示す．これらのグラフで，最も右に位置するプロットは，文字数最大しりとり問題で得られた最大文字数におけるしりとりの最長単語数を表す．そして，右から 4 番目のプロットは，最長しりとり問題の解としての文字数とそれに対するしりとり長を表している．

図 7 と図 8 から，固定文字数最大しりとり問題において，単語数が最大となるしりとりが得られる文字数には幅があることが分かる．これは，図 4 と図 5 の固定単語長文字数最大しりとり問題では顕著に見られなかった特徴である．最長しりとり問題で得られるしりとりの解は，しりとりの文字数に対して一定の範囲で存在し，文字数最大しりとり問題で得られるしりとりの解は，ある単語数のときに得られることを示している．これは，最長しりとり問題の解として得られるしりとりでは，そのしりとりで使われなかった単語（列）と使われた単語（列）の交換が可能なことを表している．しかし，文字数最大しりとり問題の解として得られるしりとりでは，そのしりとりで使われなかった単語（列）と使われた単語（列）の交換を行うと，しりとりの文字数が減ってしまうことを表している．つまり，同じ文字数で交換できる異った単語数の単語列が，しりとりで使われていない単語の中に少ないことを示している．

### 6.3 派生問題を解く計算時間

派生問題を用いたすべての実験では，最初の線形計画問題による緩和問題か最初の整数計画問題による緩和問題かのどちらかで最適解が得られていた．

広辞苑についての固定単語長文字数最大しりとり問題に対する図 4 を得るための時間は，110 秒〜180 秒（平均 136 秒，標準偏差 23 秒）であった．ただし，固定単語数が 81,970 単語のときは，解は得られず，終了した．そして，固定単語数が 78,108 単語のときは，24 時間動作させたが，最適解を得ることができなかった．表 2 から，文字数最大しりとり問題では，56 秒で解を得ていることを考えれば，固定単語長文字数最大しりとり問題の解を得るには，2 倍以上の計算時間が必要である．最も計算時間が短かったのは，最長しりとり問題で得られた最長しりとり長に対する解を求めたときで，このときは，線形計画問題で整数解を得ている．

ICOT 辞書についての固定単語長文字数最大しりとり問題に対する図 5 を得るための時間は，32〜81 秒（平均 46 秒，標準偏差 13 秒）であった．表 3 より，文字数最大しりとり問題の場合，18 秒で解を得ていることから，広辞苑の場合と同様に約 2〜3 倍の範囲の実行時間で解が得られる．ICOT 辞書の場合は，最長しりとり問題で得られた最長しりとり長に対する解を求めるときは線形計画問題で整数解が得られた．

広辞苑についての固定文字数最長しりとり問題に対する図 7 を得るための時間は，89〜505 秒（平均 212 秒，標準偏差 132 秒）であり，図 4 に比べ，計算時間

がかかり，ばらつきも大きい．ばらつきの大きさは，整数計画問題を解くのに必要とされる時間が要因となっている．計算時間を要するのは，固定文字数が少ないときである．計算時間は解を見つけるための手間と考えれば，固定文字数が少ない場合は多い場合に比べ，問題に与えられた条件が厳しいことになる．問題の定義より，しりとりがある文字数になるという条件と，しりとりの前後にしりとりで使われなかった単語がつながらないという条件では，派生問題において解が必ず求まる保証はなく，特に前後につながる単語がない条件により，条件にあてはまるしりとりを探すのは困難な作業であると考えられる．

ICOT 辞書についての固定文字数最長しりとり問題に対する図 8 を得るための時間は，27〜252 秒（平均 103 秒，標準偏差 84 秒）であった．これは表 3 で，最長しりとり問題の解を得る時間 17 秒に比べ計算時間が必要とされる．計算時間の点からは，ばらつきが大きく，固定文字数を少なくすることで，計算時間がかかる傾向があるが，先に述べたように，問題の解に対する条件が厳しくなったためと考えられる．

## 7. おわりに

本論文では，最長しりとり問題の一般化として文字数最大しりとり問題を導入し，厳密解法である LP ベースの分枝限定法による実験を行った．また，派生問題として，固定文字数最長しりとり問題と固定単語長文字数最大しりとり問題を扱った．文字数最大しりとり問題の緩和問題は，最長しりとり問題の緩和問題に対し，定数倍の変数が増える問題であり，41 倍の計算時間がかかる．しかしながら，現存する国語辞典の見出し語の範囲では十分解ける問題であることも分かった．

派生問題では，しりとり全体の文字数や単語数に関する条件により，解が求まらない場合が存在し，文字数最大しりとり問題の 2 倍以上の時間がかかるが，ほとんどの場合現実的な時間で解を得ることができた．派生問題の解を分析した結果，日本語の特徴として，最長しりとり問題では，文字数の一定の範囲で最長である解が存在するが，文字数最大しりとり問題では，狭い範囲の単語長でしか文字数が最大である解が得られないことを示した．そして，最長しりとりを作成するには短い単語長の単語が有効に使われ，文字数最大しりとりを作成するには長い単語長の単語が有効に使われていることを示した．

アークの重みが異なることで，文字数最大しりとり問題は，最大しりとり問題よりも，一般的な最長路問題に近い．このような特徴を使った実用的な応用としては，遺伝子のシークエンス[8] におけるフラグメントの順序決定や文法推論における最小状態決定オートマトンの生成[9] などが考えられる．これらの問題は，多量のデータの組合せ最適化の問題であり，文字数最大しりとり問題よりも多く変数を扱う必要がある思われるが，文字数最大しりとり問題を応用したモデルの開発と整数計画問題ソルバの性能向上により，将来的には整数計画問題により厳密解が求められると考えている．

**謝辞** 本研究の一部は科研費基盤研究（B）(2)（No.15300269）の補助を受けている．

## 参考文献

1) Ito, T., Tanaka, T., Hu, Z. and Takeuchi, M.: An Analysis of Word Chain Games, 情報処理学会誌，Vol.43, No.10 (2002).
2) Kanasugi, T., Matsuzawa, K. and Kasahara, K.: Applications of ABOUT Reasoning to Solving Wordplays, *TR. IEICE*, NLC96-31, pp.1–8 (1996).
3) Lai, H-J.: Eulerian Subgraphs Containing given Edges, *Discrete Mathematics*, Vol.230, pp.63–69 (2001).
4) Niimura, I., (Eds): Koujien Ver.4, Iwanami (1992).
5) Institute for New Generation Computer Technology: Keitaiso Jisyo, ICOT Freeware No.33 (1993).
6) Doerr, B.: Linear Discrepancy of Basic Totally Unimodular Matrics, *The Electronic Journal of Combinatorics*, Vol.7, pp.1–4 (2000).
7) Inui, N., Shinano, Y., Kounoike, Y. and Kotani, Y.: Solving the Longest Shiritori Problem, 03-MPS-48, IPSJ (2003).
8) Myers, G.: Whole-Genome DNA Sequencing, *IEEE Computational Engineering and Science*, Vol.3, No.1, pp.33–43 (1999).
9) Oliveira, A.L. and Edwards, S.: Limits of exact algorithms for inference of minimum size finite state machines, *Algorithmic Learning Theory* (*7th International Workshop*), pp.59–66 (1996).

(平成 16 年 8 月 16 日受付)
(平成 17 年 3 月 16 日再受付)
(平成 17 年 3 月 24 日採録)

**乾　伸雄**（正会員）

1963 年生．1987 年東京農工大学大学院工学研究科修了．1991 年より東京農工大学工学部助手．人工知能，自然言語処理の研究に従事．人工知能学会，IEEE 各会員

**品野　勇治**（正会員）

1961 年生．1997 年東京理科大学大学院工学研究科博士課程修了，博士（工学）．同年東京理科大学助手．1999 年東京農工大学講師（現職）．主に，数理計画法の理論と応用の研究，組合せ最適化問題に対する並列・分散アルゴリズムとその実装に関する研究に従事．オペレーションズ・リサーチ学会，IEEE，ACM 各会員

**小谷　善行**（正会員）

1949 年生．1977 年東京大学大学院博士課程修了．同年東京農工大学勤務．現在，同大学教授．人工知能，知識処理自然言語処理，知識獲得，ゲームシステム，教育工学の研究に従事．電子情報通信学会，人工知能学会等各会員．前コンピュータ将棋協会会長．