最長しりとり問題の解法

乾 伸雄[†] 品野勇治[†]
鴻池祐輔[††] 小谷善行[†]

本論文では，最長しりとり問題をネットワークの問題としてモデル化し，整数計画問題として定式化を行う．この定式化では，変数の数が頂点数に対して，指数オーダで増加するため，事実上，整数計画問題として直接的に解くことは難しい．そのため，緩和問題を設定し，LP ベースの分枝限定法によって解決した．これによって，19 万語程度の辞書から最長しりとりを Xeon2.8 GHz プロセッサの PC を使って 1 秒程度で作成することができた．また，本論文では，局所探索による解法と比較し，問題の困難さを実験的に調べた．さらに，様々なインスタンスにおける解を分析することで，最長しりとり問題の性質を調べた．

Solving the Longest Shiritori Problem

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

This paper describes the definition of the longest Shiritori problem as a problem of network flow and the solution using the the integer problem. This formulation requires a large number of variables being of exponential order. To overcome the difficulty, we propose a solution based on the LP-based branch-and-bound method, which solves the relaxation problems repeatedly and enumerates all the solutions implicitly. This method is able to calculate the longest Shiritori sequences for 190 thousand words dictionary in a second in Xeon 2.8 GHz PC. In this paper, we compare the performances for the heuristic local search and investigate the results for a variety of instances to explore characteristics of the longest Shiritori problem.

1. はじめに

ゲームとしてのしりとりは，一般には回答順番の決められた N 人で行われ，回答が求められた人は，前の人の言った単語の末尾の文字で始まる単語を回答する（本論文では，以下，単語の最初の文字を**開始文字**，単語の末尾の文字を**終了文字**とよぶ）．回答が求められた人は，ゲーム中すでに使われた単語は使用できない．回答に際しては，相手が知らない単語を言っても，その単語の存在を明らかにできれば問題がないため，しりとりは一般に不完全情報ゲームである．過去にコンピュータを用いてしりとりを扱った研究としては，概念ベースを使った連想によるしりとりの研究[2] や，しりとりの完全解を求める研究[1] などがある．

通常のしりとりは，終了文字と開始文字が一致する単語でつないでいくが，連想によるしりとりは，単語の概念によって連想される単語でつないでいく．連想によるしりとりの研究[2] では，コンピュータと人間が対戦するシステムを構築しており，コンピュータは人間により提示された単語から関係のある概念に属する単語を検索して出力する．このとき，関係のある概念からの単語の選び方は無作為であり，人間側の知識では連想しにくい単語が出たときなどには，ゲームが早く終了してしまう危険性がある．人間がゲームを楽むためにはコンピュータが連想しやすい単語を出していく工夫が必要と考えられる．

一方，完全解を求める研究として，しりとりに使える単語の集合（本論文では以下**辞書**とよぶ）が与えられている場合に，二人完全情報ゲームの枠組みでモデル化し，先手必勝か後手必勝かを判定した研究[1] がある．しかし，この研究においては，単語数の少ない辞書でもゲーム木の規模が大きくなることから，単語数の多い大規模な辞書を使った解を求めるには至っていない．

† 東京農工大学大学院共生科学技術研究部
Institute of Symbiotic Science and Technology, Tokyo University of Agriculture and Technology
†† 東京農工大学工学教育部
Graduate School of Engineeering, Tokyo University of Agriculture and Technology

しりとりを題材とした研究の場合，コンピュータ側で長くゲームを継続する，つまり人間がなかなか行き詰まらないで単語を出せるような工夫をすることで，人間にとって興味深いシステムの実現が可能となる．本論文では，しりとりを題材とした研究に対する基礎データを提供するために，最長しりとり問題を定義し，その解法を提案する．

最長しりとり問題を，次のように定義する．

最長しりとり問題
しりとりは，ある単語から始めて，その単語の終了文字が次の単語の開始文字と一致するように単語を提示していき，次に提示する単語がない場合に終了するゲームである．同じ単語が 2 度以上しりとりに現れてはいけない．本論文では，単語はすべてひらがな表記されているものとする．最長しりとり問題は，与えられた辞書において，しりとりとなる最も単語数の多い単語列を作成する問題と定義する．つまり，最長しりとり問題における “長さ” は単語数として定義する．

この最長しりとり問題に対し，次のようないくつかの表現方法が考えられる．

単語を頂点としたグラフ表現：最長しりとり問題は，単語を頂点とし，しりとりの単語列が構成可能な単語間に有向辺を持つグラフ上の最長経路問題に帰着できる．

1 文字を頂点とした木表現：ある文字からしりとりを構成する最長しりとり問題は，**図 1** に示すような文字を頂点とし，頂点間の枝を単語とした探索木上の根から葉への経路の中で最長路を求める問題としてモデル化できる．

最長経路問題は，一般に NP–困難な問題である．高速に最長経路を求める研究は並列計算の分野でよく行われている[3]．最長経路を使った実用的な研究[6] も行われているが，最短経路問題に比べ研究の数は少ない．最長経路を求めるアルゴリズムとしては，特殊な構造を持つグラフについての研究が行われている．Directed Acyclic Graph は，頂点数 n，有向辺数 m に対し，$O(n+m)$ 時間で解けることが知られている．ポジティブなサイクルを持たない Directed Cyclic Graph については，Liao-Wong Algorithm や Bellman-Ford Algorithm などが知られている．しかし，最長しりとり問題に対しては，これらのアルゴリズムを適用することは困難である．

木表現は一般的な表現方法である．たとえば，使われる単語が既知である二人完全情報ゲームにおいては，AND/OR 木となる．この場合，網羅的に探索することになり，文字数 n，最初と最後の文字が同じ単語の数の最大値を m としたとき，1 つの頂点から出る枝の種類数は $O(n)$，深さは $O(mn^2)$ であることが知られている[1]．これは，木表現に対し，網羅的な探索で最長しりとりを求めようとしても，現実的な時間内に解が得られないことを示している．

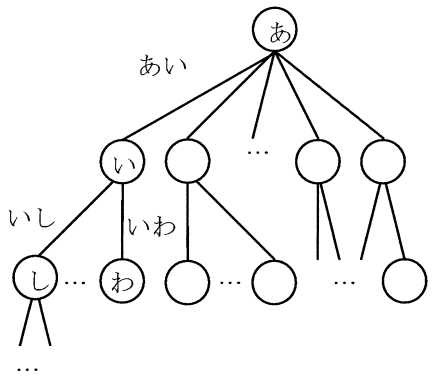
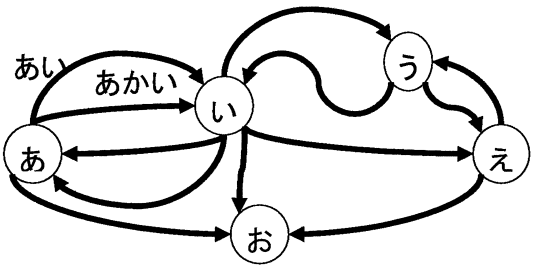


図 1 探索木によるしりとりの表現
Fig. 1 Representation of Shiritori as search problem.

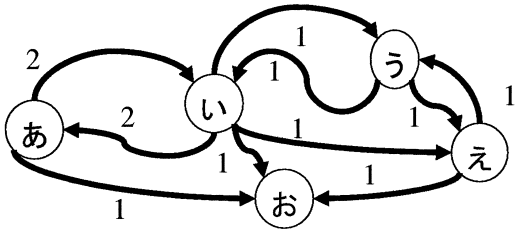
本論文では，最長しりとり問題を整数計画問題に帰着している．最長経路問題の整数計画法による解法の先行研究としては文献 4) があげられる．本研究では，最長しりとり問題を単語を頂点としたグラフ上の最長経路問題へ帰着するのではなく，この先行研究と同様の解法が適用できるようにひらがなを頂点としたグラフ表現によりモデル化を行った．2 章では提案する最長しりとり問題のモデル化を行い，3 章では最長しりとり問題を整数計画問題として定式化する．4 章では，最長しりとり問題に対する LP（Linear Programming）ベースの分枝限定法による解法を示し，5 章では局所探索によるヒューリステリック解法を示す．6 章では種々の実験結果を示し考察する．

2. 最長しりとり問題のモデル化

本論文では，与えられた辞書における単語間の関係を有向グラフとして表現し，1 つのしりとりをオイラー路（すべての有向辺をちょうど 1 回だけ使う路）に対応付け，辞書から構成される有向グラフにおける最大準オイラー部分グラフ（準オイラーグラフはオイラー閉路は持たないがオイラー路を持つグラフ）を求める問題としてモデル化する．また，その有向グラフの頂点間の多重有向辺を取り除き，多重有向辺の数を有向辺の容量としたネットワーク上でのネットワークフロー問題としてのモデル化も示す．



(a) 単語が有向辺に対応する場合



(b) 単語数が有向辺の容量に対応する場合

図 2 しりとりのモデル
Fig. 2 Network models of Shiritori.

しりとりでは，各単語の開始文字と終了文字だけが問題であるので，開始文字または終了文字になる文字集合を頂点集合 V とおく．ここでは，文字を数字に対応づけて表現し，文字数 n の集合を $V = \{1, 2, \ldots, n\}$ と表す．有向辺の集合 A は単語の集合であって，$A = B_{11} \cup B_{12} \cup \ldots \cup B_{nn}$ と表される．ここで，B_{ij} は文字 i で始まり，文字 j で終わる単語の集合である．これによって，得られる有向グラフを**図 2** (a) に示す．図 2 (a) において，すべての有向辺をたかだか 1 度だけ使うオイラー路が 1 つのしりとりを表現する．このグラフにおいて，最長しりとり問題は次のようにモデル化される．

最長しりとり問題のグラフ表現によるモデル
有向グラフ $G = (V, A)$ の準オイラー部分グラフ（Semi-Eulerian Subgraph）の中で有向辺の数が最大となるものを見つける問題．

よって，最長しりとり問題は最大オイラー路部分グラフ問題と等価であるといえる．一般に最大オイラー路部分グラフ（Largest Eulerian Subgraph）を求める問題は，最長のハミルトン路を求める問題と等しく，やはり NP-困難な問題である[5),7)]．

しりとりの長さに着目する際には，単語の違いは問題ではなく，2 つの文字を結ぶ単語の数だけを扱えば十分である．このため，文字 i から文字 j への有向辺の集合 B_{ij} に含まれる有向辺を 1 つの有向辺 a_{ij} で置き換えた有向辺の集合 $A' = \{a_{11}, a_{12}, \ldots, a_{nn}\}$ と，単語数 $f_{ij} = |B_{ij}|$ の集合 $F = \{f_{11}, f_{12}, \ldots, f_{nn}\}$ を用いたネットワークを考える．ここで，f_{ij} は有向辺 a_{ij} の容量と考える．このネットワークの例を図 2 (b) に示す．このネットワークにおいて，最長しりとり問題は次のようにモデル化される．

最長しりとり問題のネットワーク表現によるモデル
ネットワーク $N = (G = (V, A'), F)$ において，任意の 2 頂点間を流れるフローの中で，各有向辺を流れるフロー量の総和を最大化する問題．

このネットワーク表現を用いて，次章では整数計画問題として最長しりとり問題を定式化する．

3. 最長しりとり問題の定式化

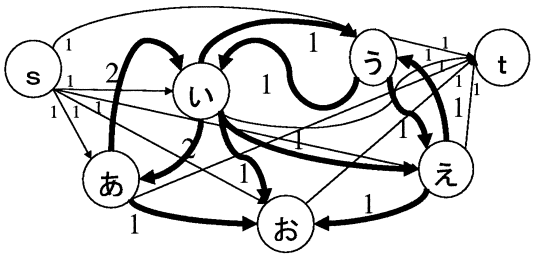
ここでは，任意の 2 頂点間のフローを考える代わりに，**図 3** に示すような補助ネットワークを作り，補助ネットワーク N' 上での $s-t$ フローを考える．補助ネットワーク N' は，ネットワーク N に対して，スーパ・ソース s とスーパ・シンク t を付加したものである．また，頂点 s から N のすべての頂点へ，および N のすべての頂点から頂点 t へ，容量 1 の有向辺を加える．補助ネットワーク N' 上で求まる頂点 s から頂点 t への最長しりとりの長さは，[本来の長さ +2] であるが，最終的に解が求まった後に，しりとりの長さを -2 すれば済むので，途中の議論では -2 を省略する．

開始文字 i，終了文字 j で，しりとりに含まれる単語の数を示す変数 x_{ij}（i，j 間のフロー量を示す）を用いて，最長しりとり問題を整数計画問題として問題 **(P)** のように定式化する．問題 **(P)** において，S_k は文字集合 V の部分集合を表す．つまり，S_k は，$S_k \subset V \cup \{s, t\}$，$S_k \neq \emptyset$，$S_k \neq V \cup \{s, t\}$ である．

変数 h_k^i は S_k 内の頂点 i に対し定義され，ある $j \in S_k$ について，$x_{ij} > 0$ または $x_{ji} > 0$ となるフローが存在するとき 0 となる $0-1$ 変数である．変数 g_k は S_k に対し定義され，S_k 内のすべての頂点について，$h_k^i = 0$ ならば 0 となる $0-1$ 変数である．つまり，S_k 内のすべての頂点が S_k 内でフローを持つならば 0 となる．変数 y_k は S_k に対し定義され，1 つの k について 0 となる $0-1$ 変数である．

問題 **(P)** の条件は，

(1) 頂点 s，t を除き，各頂点に入るフロー量と出



s より出る、および t に入る有向辺の容量 1、その他の有向辺の容量は辞書中の単語数で決定される。

図 3 補助ネットワークによるグラフ表現
Fig. 3 Graph representation using support network.

るフロー量が等しい．頂点 s では出ていくフロー量が 1 多く，頂点 t では入ってくるフロー量が 1 多い．

(2) フロー量が正の有向辺により誘導される部分グラフが連結であること

という 2 つ条件の記述からなる．

$$
\begin{aligned}
\textbf{(P)}\ \text{最大化}\ z = & \sum_{i\in V\cup\{s\},j\in V\cup\{t\}} x_{ij} \\
\text{条件}\ & \sum_{j\in V} x_{sj} = 1, \\
& \sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 0 \quad \forall i \in V, \\
& \sum_{j\in V} x_{jt} = 1, \\
& \sum_{k=1}^{2^{|V\cup\{s,t\}|}-2} (1-y_k) = 1, \\
& \sum_{\substack{i\in S_k,\\ j\in V\cup\{s,t\}\setminus S_k}} x_{ij} \geq y_k - g_k, \quad \forall S_k, \\
& h_k^i \sum_{j\in S_k} (f_{ij}+f_{ji}) + \sum_{j\in S_k} (x_{ij}+x_{ji}) \\
& \qquad \leq \sum_{j\in S_k} (f_{ij}+f_{ji}), \forall S_k, \forall i \in S_k, \\
& \sum_{i\in S_k} h_k^i - g_k \geq 0, \forall S_k, \\
& 0 \leq x_{ij} \leq f_{ij}, \quad \forall i \in V, \forall j \in V, \\
& 0 \leq x_{sj} \leq 1, \quad \forall j \in V, \\
& 0 \leq x_{jt} \leq 1, \quad \forall j \in V, \\
& x_{ij} \in \mathbf{Z}, \quad \forall i \in V\cup\{s\}, \\
& \qquad\qquad \forall j \in V\cup\{t\}, \\
& y_k \in \{0,1\}, \quad \forall S_k, \\
& g_k \in \{0,1\}, \quad \forall S_k, \\
& h_k^i \in \{0,1\}, \quad \forall S_k, \forall i \in S_k.
\end{aligned}
$$

問題 **(P)** では条件 (2) を，次のような頂点の部分集合 S_k を一意に決定することで記述している．すなわち，S_k 内のすべての頂点に接続する有向辺にフローがあり，S_k 内の頂点と $(V\cup\{s,t\})\setminus S_k$ を結ぶ頂点間の有向辺にはフローがないような集合 S_k を一意に決定することと記述している．これは，問題 **(P)** の記述では，変数 y_k を使って表現されている．y_k は，S_k から外部の頂点へフローがなく，S_k 内のすべての頂点が S_k 内にフローを持つ場合 0 となる $0-1$ 変数である．y_k がある k についてだけ 0 となる場合，$s-t$ フロー以外にフローを持つ頂点は存在しない．

問題 **(P)** において，開始文字か終了文字になる文字の数を n としたとき，(1) の条件については $(n+2)$ 個の制約式であるが，(2) は，n 個の頂点を空でない 2 つの部分集合に分ける $O(2^n)$ の制約式，変数が必要となるため，現実的に問題 **(P)** で最長しりとりを求めることは難しい．

4. LP ベースの分枝限定法による解法

前章では，最長しりとり問題を整数計画問題 **(P)** として定式化した．その定式化では，変数および式の数が指数オーダのため，整数計画問題ソルバへの入力データとして記述できるのはきわめて小規模な場合だけである．そこで，本章では緩和問題を繰り返し解くことにより，最長しりとり問題を解決する方法を提案する．最初にアルゴリズムの概略を説明し，その後，アルゴリズムの詳細を示す．

まず，フローとしての条件だけを考慮し，各有向辺を流れるフロー量の総和を最大化する緩和問題 $(\mathbf{RP}_0)$ を考える．なお一般に，緩和問題 $(\mathbf{RP}_k)$ は，緩和問題 $(\mathbf{RP}_{k-1})$ に条件を加えたものである．

$$
\begin{aligned}
(\mathbf{RP}_0)\ \text{最大化}\ z_0 = & \sum_{i\in V\cup\{s\},j\in V\cup\{t\}} x_{ij} \\
\text{条件}\ & \sum_{j\in V} x_{sj} = 1, \\
& \sum_{j\in V} x_{ij} - \sum_{j\in V} x_{ji} = 0 \quad \forall i \in V, \\
& \sum_{j\in V} x_{jt} = 1, \\
& 0 \leq x_{ij} \leq f_{ij}, \quad \forall i \in V, \forall j \in V, \\
& 0 \leq x_{sj} \leq 1, \quad \forall j \in V, \\
& 0 \leq x_{jt} \leq 1, \quad \forall j \in V.
\end{aligned}
$$

問題 $(\mathbf{RP}_0)$ は，最長しりとり問題 **(P)** の緩和問題となっているので，その目的関数値 z_0 は最長しりとり問題の上界値を与える．問題 $(\mathbf{RP}_0)$ の制約条件

は，よく知られた整数定理の成り立つ制約条件であり，各 f_{ij} が整数なので，問題 **(RP$_0$)** の最適基底解 x_{ij} は整数解として求まる．よって，仮に，**(RP$_0$)** の解 $x_{ij} > 0$ の有向辺により誘導されるグラフが連結である場合には，x_{ij} は問題 **(P)** の最適解となる．つまり，最長しりとり問題は解けたことになる．

そこで，**(RP$_0$)** の解 $x_{ij} > 0$ の有向辺で誘導されるグラフが，複数の連結成分に分かれた場合について考える．**(RP$_0$)** の制約条件より，頂点 s と頂点 t の両方を含む連結成分が必ず存在する．いま，s，t を含む連結成分の頂点集合を V_0^* とし，最長しりとり問題の許容解集合を 2 つに分割**（分枝操作）**する以下の問題を考える．

(1) 頂点 $i \in V_0^*$ から頂点 $j \in V \backslash V_0^*$ への単語の遷移が少なくとも 1 つはある最長しりとり問題

(2) 頂点 $i \in V_0^*$ から頂点 $j \in V \backslash V_0^*$ への単語の遷移は 1 つもない最長しりとり問題

この 2 つの問題の最長しりとりのうち，長い方が元の最長しりとり問題の最適解である．

(2) は，**(RP$_0$)** の頂点集合 $V \cup \{s,t\}$ を V_0^* に制限した問題に相当する．よって，新たな問題を解くことなく，**(RP$_0$)** の解から，$x_{ij}, \forall i \in V_0^*, \forall j \in V_0^*$ を取り出せば構成できる．このときの目的関数値 z_0' は，

$$z_0' = \sum_{i \in V_0^*, j \in V_0^*} x_{ij}$$

として求まる（問題 **(RP$_k$)** に対しては，$z_k' = \sum_{i \in V_k^*, j \in V_k^*} x_{ij}$）．$z_0'$ は，問題 **(P)** の許容解の目的関数値なので，最長しりとり問題の最適値の下界値を与える．

(1) に関しては，「頂点 $i \in V_0^*$ から頂点 $j \in V \backslash V_0^*$ への単語の遷移が少なくとも 1 つはある」という条件

$$\sum_{i \in V_0^*, j \in V \backslash V_0^*} x_{ij} \geq 1$$

を **(RP$_0$)** に加えて得られる問題 **(RP$_1$)** を考える（例として，**図 4** を参照）．**(RP$_1$)** を解き，同様の分枝操作を行う．以上を再帰的に繰り返しアルゴリズムを構成する．k 回目の再帰で解く問題 **(RP$_k$)** は，次のようになる．

$$\begin{aligned}
\textbf{(RP}_k\textbf{)} \text{ 最大化 } & z_k = \sum_{i \in V \cup \{s\}, j \in V \cup \{t\}} x_{ij} \\
\text{条件 } & \sum_{j \in V} x_{sj} = 1, \\
& \sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 0 \quad \forall i \in V, \\
& \sum_{j \in V} x_{jt} = 1, \\
& \sum_{\substack{i \in V_l^*, \\ j \in V \backslash V_l^*}} x_{ij} \geq 1, \ l = 0, \ldots, k-1, \\
& 0 \leq x_{ij} \leq f_{ij}, \quad \forall i \in V, \forall j \in V, \\
& 0 \leq x_{sj} \leq 1, \quad \forall j \in V, \\
& 0 \leq x_{jt} \leq 1, \quad \forall j \in V, \\
& x_{ij} \in \mathbf{Z}, \quad \forall i \in V \cup \{s\}, \\
& \qquad\qquad \forall j \in V \cup \{t\}.
\end{aligned}$$

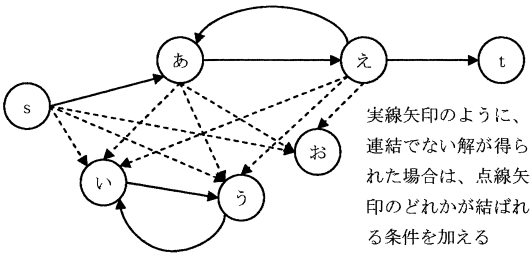


図 4 分枝操作による追加条件の例
Fig. 4 Example of supplementary condition by branch operation.

V_l^* は，**(RP$_l$)** の解における s，t を含む連結成分の頂点集合である．**(RP$_k$)** においては，整数条件を取り除くと，最適基底解の整数性は保証されない．

ここで提案している分枝限定法では，整数計画問題を緩和問題とし，さらに制約式を加えて解いている．しかし，整数計画問題ソルバの中では，線形計画問題に基づく分枝限定法が動作しているため，本論文では LP ベースの分枝限定法に基づくアルゴリズムとよぶことにする．

次に，アルゴリズムを示す．

Algorithm Making Longest Shiritori
begin
  $k := 0$;
  $z^* := 0$; { z^*:暫定値 }
  { $\mathbf{x}^*$ へ暫定解を保存する }
  while true **do**
  begin
    (RP$_k$) を解く;
    if (RP$_k$) に実行可能解がない **then**
      goto 1;
    if (RP$_k$) の解で $x_{ij} > 0$ の有向辺で構成されるグラフは連結 **then**
    begin
      if $z^* < z_k$ **then**
      begin
        $z^* := z_k$;
        z_k を導いた解を $\mathbf{x}^*$ へ保存する;
      end;
      goto 1

```
      end
    else
      begin
        if z_k < z* then
          goto 1;
        if z* < z'_k then
        begin
          z* := z'_k;
          z'_k を導いた解を x* へ保存する
        end;
        V*_k を抽出し，新しい制約条件を
        追加して，(RP_{k+1}) を作成する;
        k := k + 1;
      end
  end;
1: x* よりしりとりを構成する;
  { しりとり長は，z* − 2 }
end.
```

整数計画問題 **(P)** の最適解が LP ベースの分枝限定法によって求められれば，そこから容易に実際のオイラー路，すなわちしりとりを求めることができる．オイラー路をリストとして抽出するアルゴリズムを次に示す．各変数は次のことを示す．

$\mathbf{x}$：最適解となる単語数 x_{ij} のベクトル表現

$\mathbf{x} = (x_{11}, \ldots, x_{nn}, x_{s1}, \ldots, x_{sn}, x_{1t}, \ldots, x_{nt})$

$\mathbf{l_i}$：i 番目のしりとり列のベクトル表現

$\mathbf{l_i} = (l^i_{11}, \ldots, l^i_{nn}, l^i_{s1}, \ldots, l^i_{sn}, l^i_{1t}, \ldots, l^i_{nt})$

L：$\mathbf{l_i}$ の集合

$IN[1 \ldots n]$：頂点がチェックされたかどうかを表す配列

```
Algorithm Making Route of Shiritori(x)
begin
  L:={};
  ExtractAllCycle(x, L);
  { x には， s から t への路が残る }
  x より，しりとりを表す頂点の
    リスト r を作成;
  r_path := r の頂点集合;
  foreach i ∈ V do IN[i] = false;
  foreach j ∈ r_path do
    if IN[j] = false then
    begin
      foreach l_i ∈ L do
        if l^i_jk > 0 or l^i_kj > 0 then
        begin
          l_i を頂点のリストに変換し
            r へ挿入する;
          L := L\{l_i}
        end;
        IN[j] = true;
      end;
    r を出力する;
  end.
```

関数 *ExtractAllCycle* は，$\mathbf{x}$ より**初等的な閉路**（同じ頂点を 2 度以上通らない閉路）を取り出し，L に追加する関数である．この初等的な閉路は縦型探索を使って求められる．アルゴリズム中のベクトル $\mathbf{d} = (d_1, \ldots, d_n)$ は，個々の頂点に入る単語の数を保持するベクトルである．

```
Function ExtractAllCycle(x, L)
begin
  foreach i ∈ V do d_i = Σ_{j∈V} x_ij;
  k = 1;
  while max_{i∈V} d_i > 1 do
  begin
    foreach i ∈ V do
      while i を起点として長さ k の
        初等的な閉路 l が x にある do
      begin
        l を L に追加する;
        x より l で使われた単語を減らす;
        d より l で使われた単語を減らす
      end;
      k = k + 1
  end
end.
```

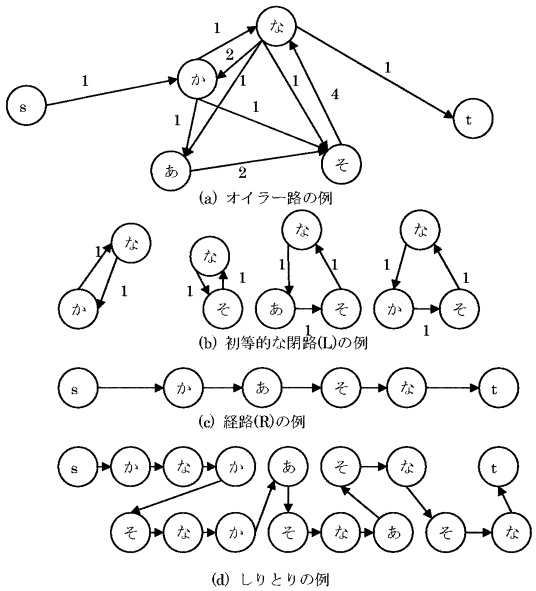


図 5 オイラー路の抽出過程
Fig. 5 Extracting process of Eulerian route.

これらの方法による，しりとりの作成過程を**図 5** に示す．一般にオイラー路上の初等的な閉路は，頂点をたどっていく仮定でバックトラックを起すことなく必ず発見できる．アルゴリズムでは省略したが，途中過程で発見した初等的な閉路を保存しておくことにより，初等的な閉路の効率的な抽出が可能になる．

5. 局所探索による解法

これまで述べてきた LP ベースの分枝限定法と比較するために，準最適解を求める局所探索に基づく解法について述べる．以下に示すアルゴリズムにおいて，$\mathbf{f}$ はしりとりに使われてない頂点 i から頂点 j への単語数 f_{ij} を要素として持ち，各要素は辞書における頂点 i から頂点 j への単語数で初期化されている．解 $\mathbf{x}$ は，しりとりに含まれる頂点 i から頂点 j への単語数 x_{ij} を要素とするベクトルとして示し，各要素の初期値は 0 である．

$\mathbf{f} = (f_{11}, \ldots, f_{nn}, f_{s1}, \ldots, f_{sn}, f_{1t}, \ldots, f_{nt}, f_{st})$
$\mathbf{x} = (x_{11}, \ldots, x_{nn}, x_{s1}, \ldots, x_{sn}, x_{1t}, \ldots, x_{nt}, x_{st})$
また，
$R(\mathbf{x}) = \{\mathbf{r}|\mathbf{x}$ 上で構成可能な，ある 2 頂点間（同じ頂点の場合も含む）を結ぶ最短のしりとりとなるベクトル $\mathbf{r}\}$.

提案する解法は，解（しりとり）$\mathbf{x}$ の長さを伸ばす部分列が $\mathbf{f}$ の中から構成可能な場合に，その部分列で $\mathbf{x}$ 内の部分列を置き換える改善方法である．

アルゴリズムの詳細は次のとおりである．ここで，アルゴリズム中の Q は，サイズ M のキューであり，サイズを超えると古いデータが消される構造を持つ．この Q を用いて，まず，過去 M 回の間に 1 度調べた開始文字，終了文字間はしばらく調べる対象から外し，解の改善を行う．過去 M 回の間に 1 度調べた開始文字，終了文字間を調べないと改善できなくなった場合には，S 上に記録されている 1 度調べた開始文字，終了文字も考慮に入れて解の改善を試みる．それでも改善しなくなった場合には，しりとりの長さが変わらないしりとり列の交換を行い，再度，改善が行われるかどうかを調べる．タブー探索と同様に，開始文字，終了文字に関する交換を抑制することで，局所解に陥りにくくしているが，解の改悪は認めていない．

Algorithm Searching Longest Shiritori
begin
 $\mathbf{f}$ 中の f_{st} へ 0 を設定;
 $x_{st} = 1$, 残りの要素は 0 により $\mathbf{x}$ を初期化;
 空のキュー Q を作成;
while true **do**
begin
 $T := R(\mathbf{x})$;
 begin
  while T が空でない **do**
  $\mathbf{r} := T$ の中の最短のしりとり;
  if $\mathbf{r}$ の始点終点が Q 内にない **then**
   if $\mathbf{f}$ において $\mathbf{r}$ より長いしりとり $\mathbf{r}'$ が構成可能 **then**
   begin
    $\mathbf{f} := \mathbf{f} - \mathbf{r}' + \mathbf{r}$;
    $\mathbf{x} := \mathbf{x} - \mathbf{r} + \mathbf{r}'$;
    $\mathbf{r}$ の始点終点を Q に入れる;
    goto 1
   end;
  $T := T \backslash \{\mathbf{r}\}$;
 end;
 $T := R(\mathbf{x})$;
 while T が空でない **do**
 begin
  $\mathbf{r} := T$ の中の最短のしりとり;
  if $\mathbf{r}$ の始点終点が Q 内に存在 **then**
   if $\mathbf{f}$ において $\mathbf{r}$ より長いしりとり $\mathbf{r}'$ が構成可能 **then**
   begin
    $\mathbf{f} := \mathbf{f} - \mathbf{r}' + \mathbf{r}$;
    $\mathbf{x} := \mathbf{x} - \mathbf{r} + \mathbf{r}'$;
    goto 1
   end;
  $T := T \backslash \{\mathbf{r}\}$
 end;
 $T := R(\mathbf{x})$;
 while T が空でない **do**
 begin
  $\mathbf{r} := T$ の中の最短のしりとり;
  if $\mathbf{r}$ の始点終点が Q 内にない **then**
   if $\mathbf{f}$ 内に $\mathbf{r}$ と同じ長さのしりとり $\mathbf{r}'$ が存在する **then**
   begin
    $\mathbf{f} := \mathbf{f} - \mathbf{r}' + \mathbf{r}$;
    $\mathbf{x} := \mathbf{x} - \mathbf{r} + \mathbf{r}'$;
    $\mathbf{r}$ の始点終点を Q に入れる;
    goto 1
   end;
  $T := T \backslash \{\mathbf{r}\}$
 end;
 goto 2;
1:
end;
2: x を出力する
end.

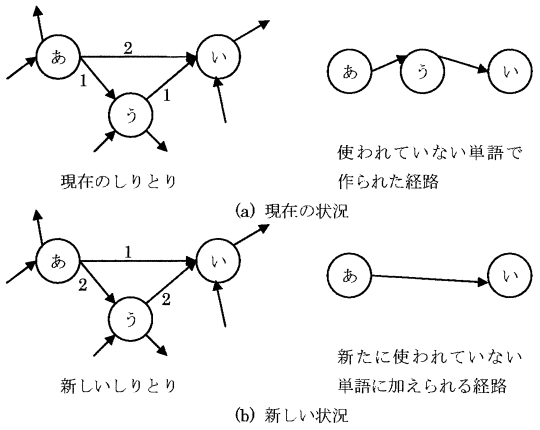


図 6 局所探索における経路の交換
Fig. 6 Exchainging routes for local heuristic search.

この局所探索の例を**図 6** に示す.

6. 実　　験

実験は数種の実在する辞書を用いた場合（6.1 節）と，開始文字，終了文字のペアが同じ単語はたかだか 1 つに限定し各文字間に単語があるかないかを恣意的に決めた仮想的な辞書を用いた場合（6.2 節）について行った．実在する辞書を用いた実験では，実際の辞書において構成される最長しりとりの長さを調べるとともに，辞書の違いにより構成できる最長しりとりの長さに，どのような違いが生じるかを調べる．また，仮想的な辞書による実験では，厳密解を求めるために用いた分枝限定法の挙動を調べることを目的とする．

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）を使用した．実験に使用した PC は，Xeon 2.8 GHz，Dual Processer，2 GB Memory である．

6.1 実在する辞書を用いた実験

一般的な辞書の場合，辞書の総単語数から見れば，辞書間で掲載されている単語の違いはそれほど大きくないと思われる．そこで，顕著な違いが生じる可能性のある以下のデータを実験に用いた．

- 広辞苑[8] より見出し語が清音で始まる単語を取り出し，見出し語の重複を省いたデータ（任意の品詞）と名詞である見出し語すべてを使ったデータ（名詞だけ）
- 中学校，高校教科書のそれぞれに掲載されている単語[9] を使ったデータ
- ICOT で開発された形態素解析用の辞書[10]（ICOT 辞書とよぶ）から名詞だけを用いたデータ

表 1 データの特徴
Table 1 Specifications of data used in experiments.

	任意の品詞	名詞だけ
単語数	137,335	192,687
文字種類数	70	70
開始文字種類数（S）	44	70
終了文字種類数（T）	70	70
有向辺数（A）	2,876	4,205
$\frac{A}{ST}$	93%	86%
有向辺数に対する最長しりとり長（L）	1,844	3,907
$\frac{L}{A}$	64%	92%

これらの見出し語に対し，最長しりとりを求めた．本実験では，しりとりをひらがなでのつながりで考えるため，末尾の記号を削除したり，旧仮名遣い，片仮名，拗音，促音を対応するひらがなに変換したりしている．たとえば，拗音「ょ」などを「よ」などに変換するといった処理を行った．

まず，広辞苑の見出し語より取り出した単語のデータの特徴を**表 1** に示す．表 1 において，文字の種類数は単語の開始文字あるいは終了文字の種類数を表す．有向辺数は頻度が 1 以上であった有向辺（開始文字と終了文字を結ぶ有向辺）数を表す．$\frac{A}{ST}$ は単語を割り付けることが可能な有向辺数に対する実際存在した有向辺数を示し，両データともかなり高い割合である．「有向辺数に対する最長しりとり長」は，有向辺の頻度を 1，すなわち，開始文字および終了文字のペアが同じである単語を一度しか使ってはいけないという条件下で解いた場合の最長しりとりの長さを表す．特に名詞だけのデータの場合は，ほとんどの有向辺を使ったしりとりが作成できている．

広辞苑を用いた実験結果を，**表 2** および **表 3** に示す．実行時間は GNU GLPK による計算時間を含む Making Logest Shiritori の実行時間で，経路を生成する時間は含まれていない．表 2 および表 3 は，開始文字と終了文字のペアが同じ単語の数を 1/2（小数点以下切捨て）ずつした辞書に対する最長しりとり問題での結果も示している．単語数（W）は，各辞書の総単語数を表し，最長の長さ（L）は最長しりとりを構成する単語数を表している．割合は，最長しりとりを構成する単語数の総単語数に対する割合を示す．IP 適用回数は，4 章で述べた分枝操作が行われた回数を示す．たとえば，IP 適用回数が 1 の場合は，$(\mathbf{RP}_0)$ で連結した最長しりとりを表す解が得られたことを示している．この回数が 2 の場合は，$(\mathbf{RP}_0)$ で連結した解が得られなかったため，$(\mathbf{RP}_1)$ は解いたが，$(\mathbf{RP}_2)$

表 2 文字頻度による最長しりとり長（任意の品詞）
Table 2 Length of Shiritori by character frequencies (all POS).

単語数 (W)	最長の長さ (L)	割合 (L/W)	IP 適用回数	計算時間 (秒)
137,335	56,519	41%	1	0.53
68,417	27,718	41%	1	0.49
33,497	13,339	40%	1	0.47
16,079	6,183	38%	1	0.39
7,406	2,654	36%	1	0.30
3,219	1,016	32%	1	0.20
1,265	303	24%	1	0.16
444	70	16%	1	0.13
124	15	12%	1	0.11

表 3 文字頻度による最長しりとり長（名詞だけ）
Table 3 Length of Shiritori by character frequencies (noun).

単語数 (W)	最長の長さ (L)	割合 (L/W)	IP 適用回数	計算時間 (秒)
192,687	86,788	45%	2	1.36
95,255	42,236	44%	1	1.20
46,595	20,077	43%	1	1.02
22,351	9,115	41%	3	0.84
10,361	3,792	37%	1	0.5
4,535	1,372	30%	1	0.27
1,853	401	22%	1	0.17
697	91	13%	1	0.12
231	18	8%	1	0.11

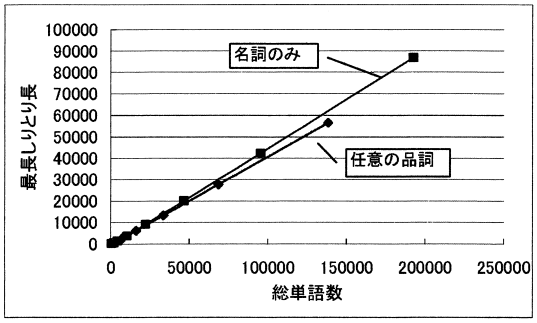


図 7 総単語数と最長しりとり長の関係
Fig. 7 Relation between words and the longest Shiritori.

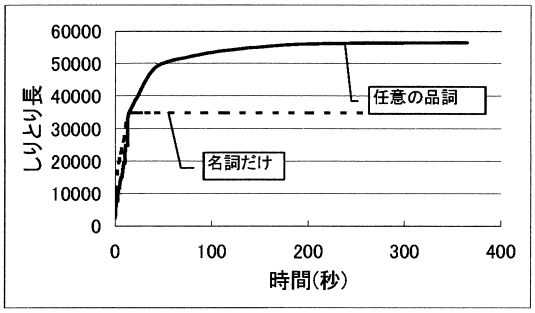


図 8 探索によるしりとり長（任意の品詞および名詞だけのデータ）
Fig. 8 Length of Shiritori by search.

以降調べられなかったことを示している．

表 2 および表 3 より，ほとんどの場合，緩和問題 $(\mathbf{RP}_0)$ を解くだけで解が求められていることが分かる．このとき，任意の品詞のデータについては，約 0.6 秒で解けており，最長しりとりの単語数は，56,519 語である．表 3 では，分枝操作が行われているデータの存在が確認できる．分枝操作が実行時間に与える影響が少なかったのは GNU GLPK の性能に依存するところであるが，ともかく，単語数に関係なく分枝操作の回数は少なかった．

図 7 は，表 2 および表 3 で示した総単語数と最長しりとり長の関係をグラフで示したものである．このグラフから，ほぼ，総単語数と最長しりとり長が比例の関係にあり，2 種類の辞書間での違いが見られないことが分かる．これより，実在の辞書では，恣意的に単語を選ばないかぎり，総単語数より最長しりとり長が予測できるといえる．

次に，前章で述べた局所探索に対する結果を示す．**図 8** は，広辞苑の任意の品詞のデータおよび名詞だけのデータに対する時間とそれまでに発見したしりとりの長さを示したものである．任意の品詞のデータに関しては M=1000 のとき，365.7 秒で最適解を得ている．しかし，名詞だけのデータについては最適解を得ることができなかった．局所探索による解法では，停止したときの解が局所解であるか最適解であるかの判断はできない．また，M の大きさによって，求まったしりとりの長さが異なる場合がある．しかしながら，このような局所探索によっても，最適解を得られる場合があることから，実在する辞書で最長しりとりを求める問題は，問題の性質として，最適解が得られやすい問題であると考えることができる．

表 4 に任意の品詞データについて，開始文字，終了文字のペアに対する長さを示す．この表では，たとえば，最大しりとり長が 56,519 単語の場合，開始・終了文字の組合せは 51 種類である．開始文字種類数は 3 で終了文字種類数は 17 であるから，その可能な組合せは 51 種類なので，この場合，すべての組合せについて同じ長さとなっている．最後の列は，開始文字，終了文字の例を示している．

この実験はスーパソース s から出る有向辺，およびスーパシンク t に入る有向辺のそれぞれについて，1 つだけの容量を 1 に，他を 0 にすることで行った．この実験結果が示すように，求められる最長しりとり長は，56,510 以上であるか，0（表には記載されていな

い 420 通りの開始文字と終了文字の組合せに対する最長しりとり長）であるかのどちらかである．また，最長しりとり長の最長と最短の違いは 9 単語であり，その差は小さい．

次に，中学校・高校教科書，ICOT 辞書を用いた結果を**表 5** に示し，広辞苑での結果である表 1 と比較する．総単語数に対する最長しりとりの長さの割合は，中学教科書の場合，広辞苑の場合とほぼ同じである．これに対して，高校教科書の場合，割合は高くない．この理由としては，高校教科書では，専門用語の出現が多くなり，特定の文字から始まる語が多くなることが考えられる．たとえば，中学教科書で，「直接」で始まる単語の数は「直接」だけなのに対し，高校教科書では 13 種類存在する．このような接頭辞の存在が，最長しりとりの長さが総単語数に占める割合を下げている要因と考えられる．ICOT 辞書（名詞だけ）と広辞苑（名詞だけ）を比較すると，全体的に顕著な違いが見られない．以上の結果から，実在する辞書の場合，どのような辞書を使っても，辞書の総単語数に対する最長しりとりの長さの割合に顕著な違いは見られないことが推測される．

表 4 始点・終点ごとの最長しりとり長（任意の品詞）
Table 4 Length of Shiritori by characters (dataset 1).

最長しりとり長	開始･終了文字ペアの種類数	開始文字種類数	終了文字種類数	開始･終了文字ペアの例
56,519	51	3	17	あ･ぐ，は･ご
56,518	197	13	26	も･ぺ，ふ･げ
56,517	389	29	35	む･ざ，ま･ぴ
56,516	424	37	53	ろ･び，り･ざ
56,515	449	38	65	ろ･で，わ･く
56,514	504	38	53	わ･れ，よ･の
56,513	395	35	44	わ･た，ま･お
56,512	194	25	35	り･せ，む･ひ
56,511	52	9	17	ろ･ほ，い･は
56,510	5	1	5	る･あ，る･へ

表 5 中学校・高校教科書，ICOT 辞書出現単語での最長しりとり長
Table 5 Length of Shiritori for words in ICOT dictionary and textbooks of junior high school and high school.

	中学教科書	高校教科書	ICOT 辞書 (名詞)
単語数（W）	7,944	38,006	110,922
文字種類数	69	69	70
開始文字種類数（S）	67	67	69
終了文字種類数（T）	69	69	70
有向辺数（A）	1,832	2,806	3,708
$\frac{A}{ST}$	40%	61%	77%
有向辺数に対する最長しりとり長（L）	1,318	2,271	3,280
$\frac{L}{A}$	71%	81%	88%
最長しりとり長（Q）	2,745	11,612	51,351
$\frac{Q}{W}$	35%	31%	46%

6.2 仮想的な辞書を用いた実験

前節の実際の辞書を用いた実験では，あまり分枝操作が実行されなかった．一般のグラフにおいてどの程度分枝操作が必要となるかを調べるため，開始文字，終了文字で使用する文字数 3〜5 までの範囲に関して，以下の条件をすべて満足するすべてのパターンの辞書群を生成し，それらのデータに対して最長しりとり問題を解いた．

- $f_{ij} = \{0, 1\}$
- $\sum_{i=1}^{n/2} \sum_{j=1}^{n} f_{ij} < \sum_{i=n/2+1}^{n} \sum_{j=1}^{n} f_{ij}$
- $\sum_{i=1}^{n} \sum_{j=1}^{n/2} f_{ij} < \sum_{i=1}^{n} \sum_{j=n/2+1}^{n} f_{ij}$
- $\sum_{i=1}^{n} \sum_{j=1}^{n-i} f_{ij} < \sum_{i=1}^{n} \sum_{j=1}^{size-i} f_{ij}$
- 最初の文字から他の文字すべてに到達する経路が存在する

これらの条件は，回転，対称，文字の入れ換えなどによる重複を排除するための条件および文字 1 からすべての文字へ到達する経路が存在することを表している．

これらのデータを解いたとき，分枝操作した回数を**表 6**，**表 7** に示す．たとえば，文字数 5 の結果である表 7 で，総有向辺数が 12 の場合を例に説明する．これは，5 頂点で可能な総有向辺数 25 本（完全グラフに自己ループを加えたもの）に対し，約半数の 12 本がフロー容量 1 の有向辺を持つ．IP 適用回数は問題を解くために必要だった緩和問題 $(\mathbf{RP}_k)$ の数である．2 段に分かれている場合，上段は最後に解いた緩和問題 $(\mathbf{RP}_k)$ で連結解が得られた場合を示す．下段がある場合では，最後に解いた緩和問題 $(\mathbf{RP}_k)$ にお

表 6 網羅的な探索における IP の適用回数（文字数 3，4）
Table 6 Applied times of IP solver for possible problems (No. of char.: 3, 4).

総有向辺数	IP 適用回数		
	文字数 3	文字数 4	
	1	1	2
4	1	0	0
5	1	25	0
6	1	40	0
7	1	291	9
8	0	194	4
9	0	655	10
10	0	204	1
11	0	328	1
12	0	49	0
13	0	41	0
14	0	2	0
15	0	1	0

表 7 網羅的な探索における IP の適用回数（文字数 5）
Table 7 Applied times of IP solver for possible problems (No. of char.: 5).

総有向辺数	IP 適用回数				
	1	2	3	4	5
6	237	6	0	0	0
7	1712	42	0	0	0
	0	1	0	0	0
8	9874	303	8	0	1
	0	4	39	11	0
9	30228	1052	31	0	0
	0	2	50	3	0
10	75220	2705	49	4	0
	0	0	240	20	2
11	141470	3434	33	0	0
	0	0	170	16	1
12	205347	4653	3	0	0
	0	0	149	15	0
13	243515	3131	3	0	0
	0	0	66	1	0
14	233535	2060	2	0	0
	0	0	19	1	0
15	183182	732	0	0	
	0	0	2	0	0
16	118727	272	0	0	0
17	64134	37	0	0	0
18	27501	8	0	0	0
19	10195	0	0	0	0
20	2759	0	0	0	0
21	626	0	0	0	0
22	103	0	0	0	0
23	9	0	0	0	0
24	1	0	0	0	0

2 段のデータ：
上段は連結解が得られた数
下段は連結解が得られなかった数
下線は制約条件によりソルバで最適解が得られなかった場合

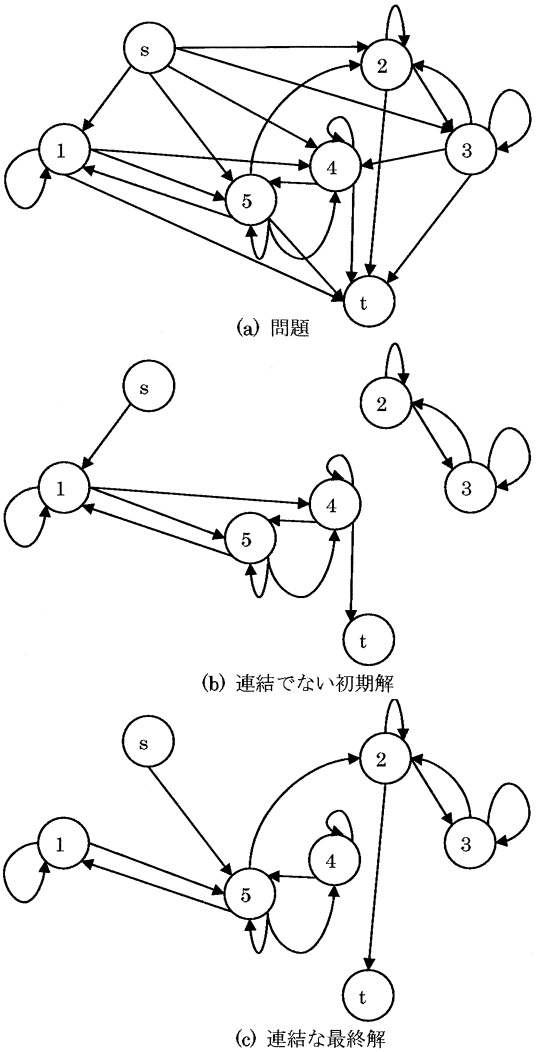


図 9 分枝操作された例（すべての有向辺の重みは 1）
Fig. 9 Example problem to which the branch-and-bound was applied (all arc weights are 1).

いて，$s-t$ フローを改善できなくて終了したか，解が存在しないために終了（下線で示される）した場合など，連結解を得ることなく終了した回数を表す．たとえば，IP 適用回数 3 の場合は，緩和問題 $(\mathbf{RP}_2)$ で連結解を得て終了した回数が 3，得ないで終了した回数が 149 となる．

自明な性質として，分枝操作は，$s-t$ フロー上にない頂点でフローが存在する場合おきる．そのため，有向辺の密度 $\left(\frac{\text{容量 0 でないフローを持つ有向辺数}}{\text{頂点数}^2}\right)$ が，高い，あるいは低い辞書では分枝操作は発生しにくい．表 6，表 7 はこのような特徴を表している．特に表 7 では，総有向辺数 11 の周辺で IP 適用回数が非常に多くなっている．なお，**図 9** に，分枝操作によって，解が改善された場合の例を示す．

表 7 に対し，文字数の少ない表 6 では，すべての場合で連結解を得ている．このような性質は辞書によるが，文字数が少ない場合でも連結解を得ることなく終了することがあるかを，次のように条件を変えて生成される辞書によって調べた．

- $f_{ij} = \{0, 1\}$
- $\sum_{i=1}^{n/2} \sum_{j=1}^{n} f_{ij} = \sum_{i=n/2+1}^{n} \sum_{j=1}^{n} f_{ij}$
- $\sum_{i=1}^{n} \sum_{j=1}^{n/2} f_{ij} = \sum_{i=1}^{n} \sum_{j=n/2+1}^{n} f_{ij}$
- $\sum_{i=1}^{n} \sum_{j=1}^{n-i} f_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{size-i} f_{ij}$
- 最初の文字から他の文字すべてに到達する経路が存在する

文字数 4 の場合の結果を**表 8** に示す．表 8 では，すべての緩和問題 $(\mathbf{RP}_k)$ において，連結解が求められなかった辞書があった．このように，小規模な問題で

表 8 網羅的な探索における IP の適用回数（文字数 4，条件を変更）
Table 8 Applied times of IP solver for possible problems (No. of char.: 5).

総有向辺数	IP 適用回数		
	1	2	3
4	21	0	0
6	154	7	1
	0	0	4
8	433	10	1
	0	0	6
10	396	3	0
	0	0	1
12	143	0	0
14	22	0	0
16	1	0	0

2 段のデータ：
上段は連結解が得られた数
下段は連結解が得られなかった数

も，単語の与え方で，最長しりとり問題の解が求められる際の過程は様々である．本節で示した結果は，ランダムに辞書を作った場合の特徴と考えられる．その結果，大部分の辞書は，最初の緩和問題 $(\mathbf{RP}_0)$ で最適解が得られることが分かった．前節で行った実在する辞書のように，有向辺の密度の高いグラフでは，連結解が得られるのは不思議なことでないといえる．

整数計画問題ソルバは，線形計画問題として得られた解が整数でない場合にだけ実行するように実装している．線型計画問題として，すべての $(\mathbf{RP}_0)$ は整数解が得られるが，$(\mathbf{RP}_1)$ 以降は整数解を得られる保証がない．しかしながら，今回行った実験の範囲では，線形計画問題を解くだけで，整数解を得ている．辞書によっては，整数解が得られない可能性は存在するが，最長しりとり問題が一般の最長経路問題とは異なる解きやすい特徴を持つことを示していると考えられる．

7. おわりに

本論文では，最長しりとり問題を定義し，その解法を示した．現在の PC の速度と実在する各種辞書に記載されている単語数に対し，リーズナブルな速度で解が求められる．より発展的な問題としては，最長文字列長しりとり問題なども考えられる．また，本研究の成果は，直接的には，人の知的好奇心をくすぐるようなシステムへの応用を考えている．対人的なシステムでは早くタスクを終了するのではなく，なるべく人間の興味を引き付けておくことが必要であり，そのようなシステムに対し，本研究の成果は活用できると考えている．

謝辞 本研究の一部は科研費基盤研究（B）(2)（No.15300269）の補助を受けました．兵庫県立大学の藤江哲也氏には，執筆にあたり，様々なコメントをいただいたことを感謝します．

参考文献

1) Ito, T., Tanaka, T., Hu, Z. and Takeuchi, M.: An Analysis of Word Chain Games, *J. IPSJ*, Vol.43 No.10 (2002).
2) Kanasugi, T., Matsuzawa, K. and Kasahara, K.: Applications of ABOUT Reasoning to Solving Wordplays, *Trans. IEICE*, NLC96-31, pp.1–8 (1996).
3) Gu, Q-P. and Takaoka, T.: A Parallel Algorith for the Longest Paths Problem on Acyclic Graphs with Integer Arc Length, *T. IPSJ*, Vol.37, No.9, pp.1631–1636 (1996).
4) Fischetti, M., Salazar-Gonzalez, J-J. and Toth, P.: *The Generalized Traveling Salesman Problem and Orienteering Problems in The Generalized Traveling Salesman Problem and its Variations*, Kluwer Academic Publisher (2002).
5) Nakayama, S. and Masuyama, S.: A Parallel Algorithm for Solving the Longest Path Problem in Outerplannar Graphs, *IEICE Trans.D-I*, Vol.J78-D-I, No.6, pp.563–568 (1995).
6) Abe, K. and Araya, S.: Train Traffic Simulation Using the Longest Path Method, *T. IPSJ*, Vol.27, No.1, pp.103–111 (1986).
7) Lai, H-J.: Eulerian Subgraphs Containing given Edges, Discrete Mathematics, 230, pp.63–69 (2001).
8) Niimura, I. (Eds): *Koujien Ver.4*, Iwanami (1992).
9) The National Institute of Japanese Language (Eds): *Chuugakkou-Koukou Kyoukasyo no Goi Chousa* (1994).
10) Institute for New Generation Computer Technology: *Keitaiso Jisyo*, ICOT Freeware, No.33 (1993).

付録 任意の品詞のデータの最長路の例

1：あヴェマリあ，あンモニあ，あンぺあ，あンプリファイあ，あンフェあ，
6：あンパイあ，あンティオキあ，あンダルシあ，あンダーウェあ，あンタキあ，
.....
56511：ようすもの，のあざみ，みあつめ，めいしいれ，れいいき，
56516：きえつく，くいら，らいはる，るモンど

(平成 16 年 2 月 4 日受付)
(平成 16 年 3 月 2 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．同年東京理科大学助手．1999 年より東京農工大学勤務．現在，同大学助教授．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．

鴻池　祐輔（正会員）

1978 年生．2003 年東京農工大学大学院工学研究科電子情報工学専攻博士前期課程修了．現在，同大学院工学研究科電子情報工学専攻博士後期課程在学中．整数計画問題の解法とその並列化に興味を持つ．日本オペレーションズ・リサーチ学会会員．

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，人工知能学会等会員．前コンピュータ将棋協会会長．