A-004

編集距離を用いたランキング集約問題と最適協調順列問題

高見 瑠莉夏[1]　徳山　豪[1,a)]

概要：ランキング集約問題とは、与えられた d 個の順列を最適に代表する順列（集約順列）を求める問題であり、WEB 検索エンジンのランキングの集約や、クラスタリング、SEO スパム検知などの応用を持つ問題である。本研究においては、2 つの順列の間の差異を編集距離を用いて定義し、それを最適集約順列を求めるアルゴリズムについての考察を行う。特に、$d \leq 3$ の場合は多項式時間アルゴリズムを与え、一般の d については近似アルゴリズムの設計を与える。また、この問題の双対問題は、順列の間の協調部分列（同時順序保持部分列）の長さあるいは重み和を類似度とした順列集合の中央値（最適協調順列）を求める問題ともとらえることができ、その見地を用いたアルゴリズムの提案も行う。

キーワード：ランキング集約問題，最大増加部分列，アルゴリズム，計算複雑度，組合せ最適化

1. 問題の導入

1.1 順列集合の代表元とランキング集約問題

複数のデータ列が与えられたとき、それらを代表する一つのデータ列を求める問題は、データ処理の典型的な問題である。この問題には、データ列の性質や、データ列の間の類似度や距離の尺度、評価基準などで様々なバリエーションがある。

例えば、それぞれのデータ列が d 次元空間の点を表すベクトルである場合は、それらの重心（平均点）や、中心点（データ点を包含する最小半径の球の中心）、中央値などが、代表するデータ列になり、統計やデータ解析では重要な概念である。また、データ列がＤＮＡの配列であれば、遺伝子の編集距離を尺度とした代表列を求める問題は、マルティプルアラインメント問題と呼ばれる、バイオインフォマティクスにおける重要な問題となる。

本研究ではデータ列として、順列を考え、d 個の順列に対して、それらを代表する順列（集約順列）を与える問題を考える。ここで、長さ m の順列とは、集合 $U = [1, n] = \{1, 2, \ldots, n\}$ の要素を m 個並べたものである。簡単のため、以下では、入力の順列は長さ n とし、U の置換とみなして積や逆置換を利用する。

集約順列問題は、複数人に対する投票の集計問題の一つの定式化であり、また、Web 検索エンジンのランキング集約問題 (Rank aggregation problem) として研究されている。投票の集計問題では一般に d は大きく、一方で Web 検索エンジンのランキング集約問題では d はランキング手法の種類であり、比較的小さい。WEB 検索のランキング集約問題においては U はサーチで検索された WEB ページの集合であり、L_i $i = 1, 2, \ldots, d$ を、サーチエンジン i による WEB ページの重要度順に並べた順列とする。この順列のことをサーチエンジン i によるランキングと呼ぶ。本来の WEB ページの重要度によるランキングは、サーチエンジンに依存しないことが望ましいが、現実にはランキングはサーチエンジンに依存し、また、サーチエンジンによっては、SEO スパム (Search Engine Optimization Spam) と呼ばれる、検索アルゴリズムの特性を悪用して、WWW 上のデータを操作して特定のページのランクを向上させる不正行為により、実際の重要度とは大きく異なるランキングが与えられることがある。

もっともシンプルな集約手法は、サーチエンジン i で k 番目にランクされたページに対して評価値 $f(i, k)$ を与え、各ページに対して d 個のサーチエンジンでの評価値の和を取り、その和の順番にページを並べたものを集約順列として利用する手法である。例えば $f(i, k) = k^{-1}$ とすると、いわゆる Zipf の法則にしたがった評価値の和に従った並べ方になる。しかしながら、この方法では、一つのサーチエンジンにおける順列での順位の大きな変更が集約順列に大きく影響し、具体的には多数のサーチエンジンでページ x がページ y より上位でありながら、y の方が集約順列では x より上位にランクされることが起こりえる。このような不公平さを制限するため、文献 [3] では、不公平さの上限を制約条件として与えた線型計画問題への定式化を行って問題を解く事が提案されている。また、Dwork ら [5] は、順

[1] 関西学院大学理工学研究科
[a)] tokuyama@kwansei.ac.jp

列の間の転倒数の総和を最小にするように最適化して、集約順列を求める定式化を提案し、$d = 4$ においてすら NP 困難であることを示し、確率的な局所探索ヒューリスティクスの提案とそれによる実験結果を与えている。

1.2 本論文の成果

1.2.1 編集コストを用いた集約順列問題の提案と定式化

本論文においては、順列の編集コストを相違度として集約順列問題を取り扱う。

要素の移動を編集の単位操作とする。すなわち、順列 $\sigma = \sigma(1), \sigma(2), \ldots, \sigma(n)$ に対して、任意の要素 $a = \sigma(i)$ を第 $\sigma(j)$ の直後の位置に移動する操作を $move(a, j)$ とする。先頭への移動は $move(a, 0)$ で表す。例として、$\sigma = 3, 5, 1, 2, 4$ に対して、$move(1, 0)$ は、1 を先頭の位置に移動する操作であり、$move(1, 0)\sigma = (1, 3, 5, 2, 4)$ となる。移動は、削除と挿入の対と考えてもよい。上記の例なら、三番目の位置にある 1 を削除し、先頭に 1 を挿入すると考えるのと同等である。

編集の最終目標の順列が σ であるとき、$[1, n]$ 上の非負の重み関数 w を与え、編集過程にある順列に対して $move(a, j)$ を施すとき、そのコストは $w(a)$ であるとする。これは、σ における要素 a の重要度を示す指標である。特に全ての重みが 1 であるとき、重みなしモデルであるという。編集において、移動した要素全体からなる集合を移動要素集合と呼ぶ。移動要素集合を X とすると、編集のコストは、X の要素の重み和 $\sum_{a \in X} w(a)$ である。順列 τ を順列 σ に編集操作の列で変換するとき、操作のコストの総和を最小にする操作列を最適変換と呼び、最適変換のコストを、$f(\tau, \sigma; w)$ と表し、τ から σ への編集距離と呼ぶ。

定義 1. 与えられた d 個の順列 $\sigma_1, \sigma_2, \ldots, \sigma_d$ と重み関数 $w_1, w_2, \ldots, w_d$ に対して、順列 τ で、$\sum_{k=1}^{d} f(\tau, \sigma_k; w_k)$ を最小にするものを、編集操作に関する最適集約順列と呼ぶ.

すなわち、最適集約順列 u は、そこから入力の d 個の置換への編集距離の総和を最小にする順列である。この問題は編集距離に関する中央値を求める問題であり、順列でなく文字列を考えると、星型アラインメント問題という名前で定式化されており、バイオインフォマティクスへの応用から考察されている [10]。文字の取り換えまで含めた編集距離においては、d が変数であるときには、NP 困難であり、近似アルゴリズムの設計においては MAX-SNP 困難である。一方、d が定数の場合は多項式時間アルゴリズムの設計が可能であり、計算量クラス XP に入る。

一方で本論文で取り扱う最適縮約順列問題は、入力は順列であり、文字列の移動、すなわち挿入と削除しか取り扱わないので、一見するとより易しい問題に見える。しかしながら、出力も順列であり、重複を許さないため、一種の巡回セールスマン問題とみなすこともでき、文字列の場合のアルゴリズムは適用できず、d が定数であっても、計算理論的に困難な問題であると予想される。

1.2.2 最適集約順列と最適協調順列問題

順列 σ を置換としてみたとき、逆置換 σ^{-1} を考える。定義から、$\sigma(i) = k$ であれば、$\sigma^{-1}(k) = i$ であり、k に対して、順列 σ で k が現れる位置、要素 k の σ における順位を与える。

2 つの長さ n の順列 σ, μ に対して、それらのジョイント列 $Y(\sigma, \mu)$ を、二次元ベクトル $(\sigma(i), \mu(i))$ $(i = 1, 2, \ldots, n)$ の列とする。インデックス集合 $A \subseteq [1, n]$ に対して、長さ n のデータ列 X に対して $X|A$ を、インデックス集合 A に対応する部分列とする。この時、次のような定義を与える。

定義 2. $Y|A(\sigma, \mu)$ が協調部分列であるとは、任意の $i \neq j \in A$ に対して $(\sigma(i) - \sigma(j))(mu(i) - \mu(j)) > 0$ が成り立つことをいう。

言い換えると、協調部分列であるとは、A において、σ の要素の大小の順位付けが μ と同じである、つまり同時部分列として同一の順序を保持していることを意味する。すなわち、$\sigma|A$ と $\mu|A$ は、順序保持マッチング [2], [6] になっている。この時、下記が成立する。

定理 1. 順列 σ と μ において、X を移動要素集合とする編集が存在するための必要十分条件は、補集合 $\bar{X} = U \setminus X$ に対して $Y|_{\bar{X}}(\sigma^{-1}, \mu^{-1})$ が協調部分列になることである。

$[1, n]$ 上の重み関数 w を与え、$A \subseteq [1, n]$ に対し、$w(A) = \sum_{i \in A} w(i)$ とするとき、順列 σ と μ の協調度 $\alpha(\sigma, \mu; w)$ を、協調部分列のインデックス集合 A で最大の重みをもつものの重みとする。

d 個の順列 $\sigma_1, \sigma_2, \ldots, \sigma_d$ と重み関数 $w_1, w_2, \ldots w_d$ に対して、順列 μ で、$\sum_{i=1}^{d} \alpha(\sigma_i, \mu, w_i)$ を最大化するものを最適協調順列と呼ぶ。これは、協調度に関して d 個のデータを代表する中央値データを与える問題である。順序保持マッチングは時系列解析や音楽情報解析などに利用されており、最適協調順列も同様に有用な概念と考えられる。

上記の定理から、次を得る。

系 1. 与えられた順列の集合 $\sigma_1, \sigma_2, \ldots, \sigma_d$ と重み関数に対して順列 μ が最適協調順列であるための必要十分条件は、$\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_d^{-1}$ と、おなじ重み関数に対して μ^{-1} が最適集約順列であることである。

したがって、最適協調順列問題と最適集約順列問題は同値な問題（双対問題）である。

1.2.3 アルゴリズムと計算複雑度

本論文では、主に小さい d の場合のアルゴリズムと計算複雑度を論じ、下記の定理を与える。

定理 2. 最適集約置換順列問題は、$d = 2$ では $O(n \log \log n)$ 時間、$d = 3$ であれば $O(n^2 \log n)$ 時間で解くことができる。

特に $d = 3$ の場合の最適集約置換は、3 つの順列の編集距離に関するシュタイナー点とみなすことができ、順列の集合のシュタイナー木や系統樹を構築するアルゴリズムの部品として利用することができる。最適集約順列問題の多

項式計算可能性は未解決であり、$d = 4$ で重みなしモデルの場合でも多項式時間で求解できるかどうかは判明していない。一方、計算困難性に関しては、重み付きで、順列の代わりに重複順列を考えて一般化した問題では、d が変数である場合は NP 困難になる事が知られている [8]。また、本論文では近似アルゴリズムの設計についても議論する。

2. 最適編集と最大増加部分列

まず、与えられた τ から σ への編集距離の計算を考察する。U の要素を σ^{-1} を施して書き換えて、$\mu = \tau\sigma^{-1}$ を単位置換に対する順列 $1, 2, 3, \ldots, n$ に編集する問題に変換できる。μ を編集するときに移動する要素の集合を $A = \{a_1, a_2, \ldots, a_s\}$ とすると、編集コストは、$\sum_{i=1}^{s} w(a_i)$ である。目的とする順列が単位置換であり、$U \setminus A$ の要素は移動されないので、μ において $U \setminus A$ の要素たちは単調増加部分列をなす。逆に、μ の単調増加部分列が与えられれば、それ以外の要素を移動させることにより、単位置換に編集することができる。したがって次を得る。

補題 1. 編集距離を計算するには、μ の最大重み単調増加部分列を計算すればよい。

順列における最大重み単調増加部分列の計算は、置換グラフの最大重み独立点集合問題と同値であり、$O(n \log\log n)$ 時間のアルゴリズムが知られている [4]。したがって、次を得る

定理 3. 長さ n の 2 つの置換の最適編集距離は $O(n \log\log n)$ 時間で計算できる。

3. $d = 2, 3$ の場合のアルゴリズム

説明の都合上、最適協調順列問題に対するアルゴリズムを与える。最適協調順列問題の入力である d 個の順列 $\sigma_1, \sigma_2, \ldots, \sigma_d$ から、行列 $M = (a_{j,k})_{1 \le j \le d, 1 \le k \le n}$ を、$a_{j,k} = \sigma_j(k)$ で定義する。M の k 番目の列ベクトルは、U の要素 k に対応する。

最適協調順列が μ で、その時の重み和を W とする。M の列を μ^{-1} で置換すると、その各行と $\iota = 1, 2, 3, \ldots, n$ との協調部分列の重み和が W と等しい。一方、ι との協調部分列は、単調増加部分列の事である。したがって、M の列の置換で、各行の最大重み単調増加部分列の重み和が最大になるものが μ^{-1} であり、これを求めればよい。

今、ある置換 τ に対して、τM の各行の最大重み単調増加列を一つ固定しよう。下記の補題は容易に確かめられる。

補題 2. τM において、もしもある列が各行の単調増加列の重み和への貢献がその列の要素の重みの最大値未満であれば、その列を適当な場所に移動して、重み和を増加させることができる。

したがって、$d = 2$ の場合は、列の要素 2 つともが単調増加列に貢献する列を最適に置換することができれば、その他の列は、最大要素が貢献するように追加していくことができる。ここで、要素が 2 つとも貢献することによる目的関数への増分は、その列の小さい方の要素重みになる。したがって、次のアルゴリズムを考える。

(1) M の第一行が昇順になるように列を置換する（すなわち、σ_1^{-1} を施す）
(2) M の第二行の最大重み単調増加列を計算する。ここで、重みは、各列の要素の小さい方の重みとする
(3) 上記の単調増加列に入らない列で、第二行に重みが大きい方の要素がある列は、第二行の単調増加列にその要素が加わって単調列になる位置に移動する
(4) 出来上がった行列の第一行の順列に対する置換を τ とし、$\sigma_1\tau$ を出力する

上記のアルゴリズムは、$d = 2$ の時の最適協調順列問題の解を与える。アルゴリズムにおいて、第二ステップは上述のように $O(n \log\log n)$ 時間で計算でき、その他のステップはバケットソートとマージ操作を用いると $O(n)$ 時間で実行できる。したがって、アルゴリズムの計算複雑度は $O(n \log\log n)$ である。なお、重みなしモデルにおいては、上記アルゴリズムで τ は単位置換になり、すなわち σ_1 自体が解であるので、計算を全くせずに問題は解ける。

$d = 3$ の時にも、補題 2 により、各列の最大重みは解における行の単調増加列に貢献できるように移動できる。したがって、M の列のうち、2 つ以上の要素が単調増加列に寄与しているもののみをまず考え、その後補題 2 を用いて残りの列を配置する。

アルゴリズム設計のために、いくつかの用語を準備する。M の列をいくつか重複を許して並べて作った行列 B と、B の第 i 行における単調増加部分列 $\mathbf{a}_i$ $(i = 1, 2, 3)$ を考えた組 $\mathbf{B} = (B, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ を候補図式 (candidate diagram) と呼ぶ。候補図式において、全ての列において、その 3 つの要素のうち少なくとも 2 個が $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$ のいずれかに含まれているとき、候補図式が二重被覆図式であると呼ぶ。

補題 3. 二重被覆図式には、M の列は重複して現れない。

証明：列 $\mathbf{c} = (c_1, c_2, c_2)$ が B に二回現れるとする。一般性を失わず、最初の出現において $c_1 \in \mathbf{a}_1$, $c_2 \in \mathbf{a}_2$ であるとしよう。すると、二回目の出現において、2 つの要素が単調増加部分列に含まれるので、どちらか、たとえば c_1 が $\mathbf{a}_1$ に含まれることになる。しかし、$\mathbf{a}_1$ は単調部分列なので、同一要素は二回現れない。したがって列が重複して現れることはない。証明終了。

二重被覆図式 $\mathbf{B}$ において、列 $\mathbf{c}$ の有効重みを、$\mathbf{c}$ において単調増加部分列に入る要素の重みの和から、$\mathbf{c}$ の要素の最大重みを引いたものとし、$\mathbf{B}$ の有効重み $\tilde{W}(\mathbf{B})$ を、そのすべての列の有効重みの和とする。M の各列の最大重みの総和を W_0 とする。定義から、次の補題を得る。

補題 4. 二重被覆図式 $\mathbf{B}$ に対して、B に入らない列を補題 2 を用いて配置すると、最適協調順列で重み和が $\tilde{W}(\mathbf{B}) + W_0$ であるものが得られる。一方、最適協調順列に対応して列

を並べ替えた行列から、単一の要素しか行の最大重み部分列に貢献しない列を除くと、二重被覆図式で、有効重みを最大にするものを得られる。

したがって、二重被覆図式で有効重みを最大にするものを求めれば、上記の補題により、最適協調順列を求めることができる。二重被覆図式においては、重複を気にせずに最適化を行えば、補題 3 から自動的に重複を排除した図式が得られることに注意しよう。

定理 4. $d = 3$ のとき、有効重みを最大にする二重被覆図式は $O(n^2 \log n)$ 時間で計算できる。

定理を証明する。まず、標準的な動的計画法によるアルゴリズムの設計指針を示し、その後計算時間を改良する工夫を述べる。インデックスの 3 つ組 $\mathbf{r} = (r_1, r_2, r_3) \in [1, n]^3$ に対して、$i = 1, 2, 3$ それぞれで第 i 行の単調増加列の最終要素が r_i である二重被覆図式の有効重み全ての最大値を $F(\mathbf{r})$ とする。もしそのような二重被覆図式が存在しなければ $F(\mathbf{r}) = 0$ とする。

$\mathbf{r}$ と 2 つ以上要素を共有する M の列 $\mathbf{c}$（高々一つしか存在しない） を考える。そのような列が存在しなければ、$F(\mathbf{r}) = 0$ である。この時、$\mathbf{c} = (c_1, c_2, c_3)$ とし、$w(c_1), w(c_2), w(c_3)$ をそれぞれの要素の重み、$w_0(\mathbf{c})$ をそれらのうちの最大の重みとする。すると、二重被覆図式の定義から、以下の再帰式が成立する。

(1) $\mathbf{c}$ が $\mathbf{r}$ と 2 つ要素を共有する ($c_1 = r_1$, $c_2 = r_2$, $c_3 \neq r_3$ とする) とき、$F(\mathbf{r}) = \max_{q_1 < r_1, q_2 < r_2} F((q_1, q_2, r_3)) + \max\{w(c_1) + w(c_2) - w_0(\mathbf{c}), 0\}$.

(2) $\mathbf{c} = \mathbf{r}$ の時、$F(\mathbf{r}) = \max_{q_1 < r_1, q_2 < r_2, q_3 < r_3} F((q_1, q_2, q_3)) + w(c_1) + w(c_2) + w(c_3) - w_0(\mathbf{c})$.

この再帰式に従って、辞書式順序で動的計画法を実行することができ、標準的な動的計画法のバックトラックで二重被覆図式が求まる。ナイーブに実装すると、$\mathbf{r}$ の候補は $O(n^3)$ あり、再帰式の計算に最悪 $O(n^3)$ の計算時間を要するので、アルゴリズムの計算時間は $O(n^6)$ となる。しかしながら、F の値が 0 でない $\mathbf{r}$ は、M の n 個の列のどれかと 2 つ以上の要素を共有し、その成分は n 以下の自然数なので、$O(n^2)$ 個しかない。

再帰式の第一のケースでの $\max_{q_1 < r_1, q_2 < r_2} F((q_1, q_2, r_3))$ は、領域 $x < r_1, y < r_2, z = r_3$ に入る (x, y, z) における F の値の最大値なので、z を固定すると、二次元の領域探索問題（詳しくは領域最大値探索問題）である。領域探索問題には、$O(\log^2 n)$ 時間で探索と、データの挿入が行えるデータ構造が知られている ([9])。第一番目の再帰式は動的計画法において $O(n^2)$ 回計算するので、計算時間は $O(n^2 \log^2 n)$ である。

再帰式の第二番目のケースの計算は、おなじデータ構造を $z = 1, 2, \ldots, r_3 - 1$ まで用いることにより $O(n \log^2 n)$ で実行できる。一方、再帰式の第二番目の計算は $O(n)$ 回しか行われない。したがって、計算時間は $O(n^2 \log^2 n)$ である。以上により、$O(n^2 \log n)$ 時間のアルゴリズムが構築できた。証明終了。

したがって、最適協調順列は $d = 3$ の時 $O(n^2 \log^2 n)$ 時間で計算できる。順列が重複順列であるときも同じ計算時間で解くことができるが、本論文では割愛する。

4. 近似アルゴリズムの設計

重みのない場合、「集約順列として d 個の置換のうちで最も良いものを選ぶ」という非常に単純なアルゴリズムが最適集約順列問題の 2 近似アルゴリズムになる。一方で、重みのある場合は、上記のアルゴリズムに対する品質保証の議論は成立しない。重みが正規化されている、すなわち M の各列の重みが同一ならば、M の重みの総和を Z とすると、下記が成立する。(以下証明は本稿では省く)。

定理 5. d が定数で、最適集約順列の編集重み和を $W = rZ$ とするとき、$(2 - r)W$ 以下の編集重み和をもつ近似解を多項式時間で計算できる。

最適協調順列は最大化問題であり、d に依存しない定数近似は容易でない。d が定数の時は、下記の結果を与える。

定理 6. d が定数で、重みが正規化されていれば、最適協調順列問題に対して最適解の重みを W_{opt} とすると $W_{alg} \geq \frac{W_{opt}}{d^{1/2}}$ かつ任意の $0 \leq \epsilon \leq 1$ に対して $W_{alg} \geq 2\epsilon W_{opt} - \epsilon^2 Z$ を満たす重み W_{alg} を持つ解を多項式時間で計算できる。

参考文献

[1] Y. Azar, L. Gamzu, X. Yin, Multiple Intents Re-ranking, *Proc. 41st ACM STOC*, :669-678, 2009.

[2] E. Cambouropoulos, M. Crochemore, C.S. Iliopoulos, L. Mouchard, Y.J. Pinzon, Algorithms for computing approximate repetitions in musical sequences, *Int. J. Comput. Math.* **79 (11)**, pp. 11351148, 2002.

[3] L.E. Celis, D. Straszak, N.K. Vishnoi, Ranking with Fairness Constraings, *Proc. 45th ICALP*, pp.28-1–28-15, 2018.

[4] M.-S. Chang, F.-H. Wang, Efficient Algorithms for the Maximum Weight Clique and Maximum Weight Independent Set Problems on Permutation Graphs, *Information Processing Letters*,**43**, pp. 291-295, 1992.

[5] C. Dwork, R. Kumar, M.Naor, D. Sivakumar, Rank Aggregation Methods for the Web, *Proc. WWW '01*, pp. 613622, 2001.

[6] J. Kim, P. Eades, R. Fleisher, S-H. Hong, C. Iliopoulos, K. Park, S. J. Puglisi, T. Tokuyama, Order Preserving Matching, *Theoretical Computer Science* **525**, pp. 68-79, 2014.

[7] 藤原直紀、徳山豪、 魔女の最適調合問題– 非減少部分列の最適化に基づく多次元ソーティング問題 情報処理学会研究報告 2023-AL-191, pp.1-8, 2023.

[8] N. Fujihara, T. Tokuyama, Sorting Columns of a Matrix to Optimize Nondecreasing Subsequences of Rows, *The 12th Japanese-Hungarian Symposium*, 2023.

[9] F. P. Preparata, M. I. Shamos, *Computational Geometry – An Introduction*, Springer Verlag 1985.

[10] L. Wang , T.Jiang, On the Complexity of Multiple Sequence Alignment, *J. Comput Biol.* 1994 Winter;1(4): pp. 337-348.