北海道大学工学部研究報告
第128号（昭和60年）

Bulletin of the Faculty of Engineering,
Hokkaido University. No.128 (1985)

二つの有限集合間の対応づけに関するアルゴリズム
―― 多重リンク型データ構造の応用 ――

伊達　惇　　野中　秀俊

（昭和60年6月30日受理）

On the Assignment Algorithm by Means of Multi-linked Data Structure

Tsutomu DA-TE and Hidetoshi NONAKA

(Received June 30, 1985)

Abstract

A problem of assigning individuals to classes under certain rules is presented. An individual can be assigned to one class according to both the priorities attached from an individual to classes and from a class to individuals. We often encounter such problems, for instance, the assignment of students to departments or certain goods to storehouses, and so on.

In the present algorithm, two-dimensionally multilinked data structure is used. The utility of the algorithm and time complexity are discussed to some extent, notwithstanding it's difficulty in making theoretical estimations for time complexity generally. Several results of computer simulations are given to illustrate the expected time complexity of the algorithm.

1．はじめに

有限個の要素を持つ二つの集合の要素間の対応づけを，ある制約条件のもとで行うという問題を考える。制約条件は次節で定義するが，一つの要素に対応づけすることが可能な他方の集合の要素の個数は，いずれの集合においても制限されていること，また対応づけのための優先順位が各集合の全要素に独立に与えられていることが基本的な特徴である。以下，便宜のために，一方を「個体」の集合，他方を「クラス」の集合と呼ぶことにする。すなわち問題は，複数の個体を複数のクラスに対応させる，という形に置き換えられる。

このような問題を考えるに至った動機は，計算機の利用目的が科学計算指向から事務処理指向へとかわり，ジョブの数，計算時間のいずれの面においてもこの傾向が強められているにもかかわらず，事務処理問題の計算機への期待の内容が高度化するにつれて，処理アルゴリズムの開発は逆に個別化が進んで手法の統一という方向が全く見られない，という現状の打開の必要性を感じた点にある。この視点は，制約条件の設定のしかたにより実に多様な問題を構成しうるので，個別アルゴリズムの微小異同を整理しつつ，手法の統一化を指向することを可能とするものと考

＊　情報工学専攻　情報科学

える。直接の応用としては，企業における人員配置，生産工程における機械と原材料の配置，商品の倉庫への割り当て，入社試験，入学試験，大学における学部移行など，さまざまな問題が想定されるが，本報告では，次節で述べるような制約条件を設定することによって，モデルとしての問題を具体化し，そのアルゴリズムを提示するとともに，データ構造として二次元的多重リンク構造を採用した場合の効率を，シミュレーションにより論ずる。

2．問題の構成およびアルゴリズム

問題を構成する基本要素は次の四つにまとめられる。

(i) 各個体が全クラスに対して設定する優先順位
(ii) 各クラスが全個体に対して設定する優先順位
(iii) 各クラス（個体）にとっての対応づけ可能な個体（クラス）の最大数
(iv) 上記の優先順位群を用いた対応づけの規則

個体の集合とクラスの集合に対して，上の四つの基本要素を制約条件として具体化したものを課したとき，複数の個体（1，2，……，n）を複数のクラス（1，2，……。m）に対応させるアルゴリズムを得ることが本節でいう問題である。(i)，(ii)で述べた優先順位は，それぞれ複数となる場合もある。たとえば，入社試験の例において，クラスに相当する会社が個体に相当する志願者に対して，筆記試験の成績による席次を一つの優先順位として設定し，面接試験あるいは信用調査結果を別の優先順位として設定するという場合がある。(iii)における最大数は，通常は一つの個体がただ一つのクラスに対応づけされる場合が多いが，二以上のクラスに対応づけされることもありうる。たとえば，人員配置問題において一人の人間が二以上の職務を兼務する場合などである。個体やクラスは必ずしも物理的存在である必要はなく，物の属性や概念のような抽象的存在であっても差し支えない。

以上の枠組の中で，次のような具体的な問題を考える。ある個体があるクラスに対応づけされることを処理手続の途中では，受容と呼び，逆に対応づけの対象から除外されることを排除と呼ぶことにする。このとき，問題の基本要素(i)～(iv)を次の(i)′～(iv)′の形に特定する。

(i)′ 個体毎に各クラスに対する優先順位をつけ，これを各個体のクラスへの志望順位と呼ぶ。
(ii)′ クラス毎に各個体に対する優先順位をつけ，これを各クラスにおける個体の席次と呼ぶ。
(iii)′ 各個体は唯一つのクラスに対応づけされるものとし，クラスiの対応づけ可能な個体数の最大数をk_i（i＝1，2，……，m）とする。nを個体の総数として，次の仮定をする。

$$\sum_{i=1}^{m} k_i \geqq n,\quad k_i > 0 \qquad (i=1,\cdots,m)$$

(iv)′ 各クラスにおいて，受容された個体の席次が，排除された個体の席次よりも下位であってはならない，という規則のもとで，各個体を志望順位の最も高いクラスに受容させる。

［問題へのアルゴリズム］

クラスに対して適当に順番を定め，以下に示す受容処理と排除処理を各クラスに対して行う。これを反復して，すべてのクラスにおいて受容される個体も排除される個体も出ない，という状態において補充処理を実行する。たかだか一回の補充処理を行うことにより，対応づけは完了する。

(1) **受容処理**　各クラスの席次表において，席次が，上位から最大受容数の範囲にあって，かつそのクラスの志望順位を最上位としている個体を，そのクラスに受容する。また，その個体を他のクラスの席次表から抹消する。

(2) **排除処理**　各クラスの席次表において，そのクラスへの志望順位を最上位としている個体のみを上位から数えたとき，最大受容数の順番にあたる個体をpとする。pよりも席次が下位である個体をこのクラスから排除する。また，排除された各個体の志望順位表において，このクラスを抹消してこのクラスよりも下位であったクラスの志望順位を，逐次くり上げる。

(3) **補充処理**　未定の状態にあるすべての個体を，その個体の志望順位の最上位に受容させる。

［アルゴリズムの完全性］

すべての個体は，処理手続きを行う前は，受容・排除について未定の状態にある。この未定の状態にある個体が，なくなったときであってかつ，(iv)′の要件がみたされるときにアルゴリズムの目的は達成される。アルゴリズムは簡単なものが良いとすれば，たとえば受容処理のみで目的が達成されることが望ましいが，(iv)′の規則設定のもとでは，次のように補充処理は必要である。

志望順位と席次がすくみの関係になっているとき，すなわち，席次が下位となっているクラスへの志望順位を上位とする個体が二以上くみ合わせられてすくみの関係になっているとき，受容処理によってどの個体も受容されず，排除処理によってどの個体も排除されず，なおかつ未定の状態の個体が残っている，という状態が生ずる（補充処理の必要性）。

この状態にあるとき，いずれのクラスにあいても志望順位を最上位とする個体の数は，そのクラスの最大受容数をこえることはなく，また，排除されずに残っている個体の席次は，これ以前に排除された個体の席次よりもつねに上位である。したがって，補充処理を行ったときに，未定の状態にある個体はすべていずれかのクラスに受容され，受容された個体の数がそのクラスの最大受容数をこえることはなく，かつあらたに受容された個体の席次が，それ以前に排除された個体の席次よりも下位であることはない（補充処理の十分性）。

以上により，(i)′～(iv)′を基本要素とする問題へのアルゴリズムの完全性は示された。

［アルゴリズムの柔軟性］

柔軟性とは，問題の構成をかえたときに，アルゴリズムの変更の度合が少ないことをいうものとする。問題の基本要素(iii)′における最大受容数は，最初に与えたものが最後まで不変であることを前提としているが，一定の条件をみたしたときには可変である，という場合を考える。(ii)′において与えられた優先順位において，同一の席次をもつ個体がいくつもある場合に，あるクラスの受容が最下位において競合することがあり，このときに競合した個体をすべて受容する（又はすべて排除する）という規則を追加して，最大受容数を変更する場合を考えるわけである。

この場合，最大受容数の定義を次のようにkからk′へかえることにより，アルゴリズムそのものは全く変更しないで対処することができる。

一つのクラスにのみ着目して，与えられた最大受容数をk，変更後の最大受容数をk′，そのクラスへの志望順位を最上位とする個体の集合をG，席次が個体p以上である個体の集合をU_p，個体pと席次が同一の個体の集合をH_p，と名付け，さらに集合Aの要素の個数を$|A|$，補集合を$\bar{A}$

であらわすものとする。

受理処理においては $|U_p|=k$ をみたす p に対して，

$$\begin{cases} p\in G \text{ かつ } H_p\cap\overline{U_p}\neq\phi \text{ のとき} & \begin{cases} k'=k+|H_p\cap\overline{U}_p|, & (\text{競合個体を受容}) \\ k'=k-|H_p\cap U_p|, & (\text{競合個体を排除}) \end{cases} \\ \text{そうでないとき} & k'=k, \end{cases}$$

とする。

排除処理においては，$|G\cap U_q|=k$ をみたす q に対して，

競合個体をすべて受容する場合は，

$$\begin{cases} H_q\cap\overline{U}_q\neq\phi \text{ のとき} & k'=k+|H_q\cap\overline{U_q}|, \\ \text{そうでないとき} & k'=k, \end{cases}$$

競合個体をすべて排除する場合は，

$$\begin{cases} H_q\cap\overline{U_q}\neq\phi \text{ であってかつ} & \begin{cases} (H_q\cap\overline{U_q})\cap G\neq\phi \text{ のとき} & k'=k-|H_q\cap U_q|, \\ (H_q\cap\overline{U_q})\cap G=\phi \text{ のとき} & k'=k+|H_q\cap\overline{U_q}|, \end{cases} \\ \text{そうでないとき} & k'=k. \end{cases}$$

以上により，問題を構成する基本要素である最大受容数について，不変であったものを可変とする，というような変更を行ったとしても，各クラスにおいて与えられた最大受容数 k を，上記の式にしたがって，k′に変更することにより，つまり用語の解釈をかえるという変更により，アルゴリズムを全く変更せずに対応づけを完了させることが可能である。これは，本アルゴリズムの柔軟性の一面と考えることができる。

3．データ構造とアルゴリズム

クラス及び個体別レコードを，図1に示すように構成する。このようなレコードを，個体とクラスのすべての組合せの数，すなわち n×m個構成し，2節で述べたクラス別席次表及び個体別志望順位表に従って，二次元的にこれらをリンクする。例えばクラスYにおける個体A～Fの席次が，B，C，A，E，F，Dの順であったとすると，ポインタ（infer）によるリンクは図2のようになり，また，個体Bのクラス X～Z に対する志望順位が，Y，X，Z であったとすると，ポインタ（upper，lower）によるリンクは図3のようになる，ここで志望順位が最上位のクラスにおける upper 欄は nil であり，また，レコードに対するクラスへの受容は，

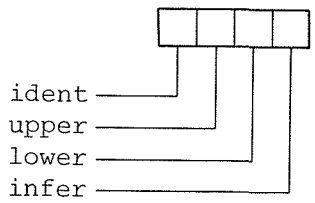

図1 レコードの構成
ident ：個体番号（1，2，……，n）
upper ：志望順位上位クラスへのポインタ
lower ：志望順位下位クラスへのポインタ
infer ：席次下位個体へのポインタ

$$\text{upper}=\text{lower}=\text{nil}$$

で定義する。表1，表2，及び図4に，クラス数3，個体数6の場合の二次元多重リンク構造の例を示す。

2節で述べたアルゴリズムを，ここで定義したデータ構造を用いて表現しなおす。アルゴリズ

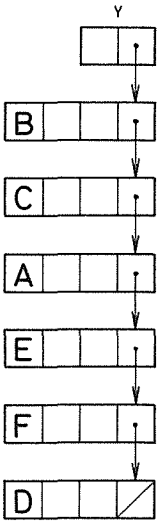

図2　席次リスト

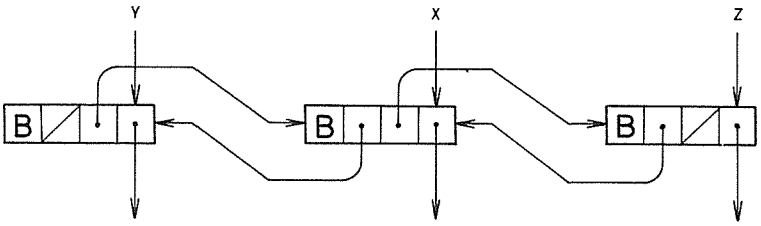

図3　志望順位リスト

表1　席次表

X	Y	Z
A	B	B
B	A	A
C	C	D
D	D	C
E	F	E
F	E	F

表2　志望順位表

A	B	C	D	E	F
X	X	Z	Z	X	Y
Y	Y	Y	Y	Y	Z
Z	Z	X	X	Z	X

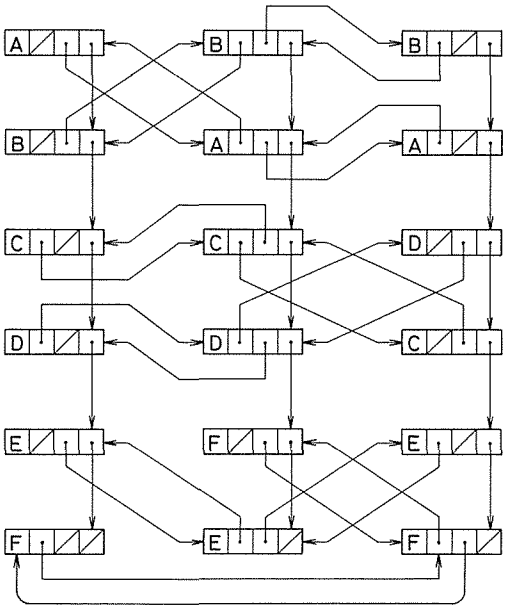

図4　多重リンク構造

ムは，以下に示す三つの処理から成る。

(1)　受容処理

a　席次リストを順に参照し，ident＝0 であるレコードを削除する。

b　上記以外のレコードについて，最大受容数の範囲内でかつ upper＝nil，lower≠nil であるレコードを受容する。

c　受容にあたっては，志望順位リストをたどり，すべてのクラスの upper 欄，lower 欄を nil にする。またさらに，受容したクラス以外のクラスの ident 欄を 0 にする。

(2) 排除処理

a 席次リストを順に参照しながら，upper＝nil であるレコードを最大受容数まで数えあげ，その最後のレコードの infer 欄を nil にする。

b 席次リストにおいて上記レコードより下位のすべてのレコードを，双方向ポインタを用いて，志望順位リストから削除する。

(3) 補充処理

upper≠nil である全てのレコードを，席次リストから削除する。

各クラスに適当な順序を定め，受容処理をと排除処理をこの順で繰返し行う。そして，どのクラスに対しても受容処理と排除処理によるリンク構造の変化が無くなったところで，補充処理を行い，手続を終える。このとき，残った席次のリスト（infer 欄によるリスト）は，最終的なクラス別被受容個体の席次リストを与える。

アルゴリズムに二次元的多重リンク構造を採用することによって次に挙げるような効用が得られる。ident＝0 の参照を除くすべての処理（受容処理，排除処理等）及びすべての表現（第一志望：upper＝nil，受容：upper＝lower＝nil）等はポインタによって表され，クラス名や個体名の情報は一切使わない。また，アルゴリズムはデータのサイズ（n×m）に依存しない。これらのことは，個体数またはクラス数の異なる問題に対し，単一のプログラムで対応できることを意味する。処理時間に関しては，まず，排除処理における志望順位の繰上げ作業が，ポインタを用いない場合は最大m－1回のデータの書き換えを必要とするのに対し，上記の方法では，upper 欄と lower 欄のそれぞれ1回の書き換えだけで終了する。また，他のクラスに受容された個体は，それ以後参照しないという点も，処理時間短縮の上で有効である。すなわち，あるクラスに個体が受容されると，その個体の他のクラスにおけるレコードは席次表から削除されるので，レコードの参照回数が減少する。特に，各個体が，席次が上位のクラスの志望順位を上位とする傾向にある場合には――この問題が応用される局面では実際このような傾向になることが多いが――上記のレコードの削除による参照回数の減少は顕著である。

4．アルゴリズムの手間に関する考察

時間計算量の尺度として，便宜上，手続が終了するまでに要する受容・排除処理の回数を用いる。受容･排除処理は，クラス x_1，x_2，……x_m を適当な順序に並べ替え，$x\sigma_{(1)}$，$x\sigma_{(2)}$，……$x\sigma_{(m)}$ の順に行うものとし，これを1サイクルと呼ぶことにする。また簡単のため次の仮定を設ける。

$$k_i = \text{const} = k \qquad (i=1,2,\cdots\cdots m), \qquad n = km$$

一般に，手続が終了するまでに要するサイクル数は，m，k 及びリンク構造に依るだけでなく，クラスの処理の順番，すなわち σ にも依存する。実際，例えば表3，表4の例では x_1，x_2……x_m の順で処理を行えば1サイクルで終了するが，x_m，x_{m-1}……x_1 の順で行えば，mサイクルを要する。このことからわかるように，時間計算量の一般的な評価は困難である。

ここでは，特に手間が最大となる場合について，サイクル数を計算する。各サイクルにおけるリンク構造の変化が一箇所だけにしか起らない場合，つまり，各サイクルで排除される個体が一個だけ出るように状況が，最悪の場合に該当する。各クラスで排除される個体は n－k 個を超えることは無いから，結局サイクル数は，

$$(n-k)m \quad = \quad km(m-1)$$

表3　席次表

X_1	X_2	X_3	- - -	X_m
A_1	A_1	A_1	- - -	A_1
A_2	A_2	A_2	- - -	A_2
⋮	⋮	⋮		⋮
A_n	A_n	A_n	- - -	A_n

表4　志望順位表

A_1	A_2	A_3	- - -	A_n
X_1	X_1	X_1	- - -	X_1
X_2	X_2	X_2	- - -	X_2
⋮	⋮	⋮		⋮
X_m	X_m	X_m	- - -	X_m

表5　席次表

X_1	X_2	X_3	- - -	X_{m-1}	X_m
A_1	A_1	A_1	- - -	A_1	A_k
⋮	⋮	⋮		⋮	
A_{k-1}	A_{k-1}	A_{k-1}	- - -	A_{k-1}	
A_k	A_{2k}	A_{3k}	- - -	$A_{(m-1)k}$	
				⋮	
				A_n	
		⋮		A_k	⋮
	⋮				
⋮		A_n			
		A_k			
	A_n			⋮	A_n
	A_k	⋮			A_1
	⋮				⋮
A_n	A_{2k-1}	A_{3k-1}	- - -	$A_{(m-1)k-1}$	A_{k-1}

表6　志望順位表

A_1 - - - A_{k-1}	A_k - - - A_{2k-1}	A_{2k} - - - A_{3k-1}	- - -	$A_{(m-1)k}$ - - - A_{mk}
X_m - - - X_m	X_2 - - - X_2	X_3 - - - X_3	- - -	X_1 - - - X_1
X_1 - - - X_1	X_3 - - - X_3	X_4 - - - X_4	- - -	X_2 - - - X_2
⋮ ⋮	⋮ ⋮	⋮ ⋮		⋮ ⋮
		X_{m-1} - - - X_{m-1}		
	X_{m-1} - - - X_{m-1}	X_1 - - - X_1		
	X_1 - - - X_1	X_2 - - - X_2		X_{m-1} - - - X_{m-1}
X_{m-1} - - - X_{m-1}	X_m - - - X_m	X_m - - - X_m		X_m - - - X_m
k-1	k	k		k+1

を超えることは無い。ここで実際，表5，表6の例において，x_m，x_{m-1}……，x_1 の順で各クラスに処理を行った場合のサイクル数は，

$$k(m-2)^2+m-1$$

となる。よって手間が最大の場合のみのサイクル数は，$O(km^2)=O(nm)$ であることが示された。一方，手間が最小の場合のサイクル数は，任意の k，mに対し明らかに 0 である。このように，

時間計算量は場合によって大幅に異なる。従って処理時間の評価には，平均的な時間計算量を用いることが必要である。前にも述べたように平均時間計算量の理論的な評価は困難であるから，いくつかのモデルに対するシミュレーションに依らざるを得ない。

5．計算機シミュレーション

アルゴリズムの平均時間計算量の評価のために，計算機のシミュレーションを行った。なお，プログラムは8086のアセンブリ言語を用いて記述した。レコードの番地は，ワード単位のオフレットで表現し，ident, upper, lower, inferの各欄をそれぞれセグメント4000 H, 5000 H, 6000H, 7000 Hに対応させた。また，各ポインタのnilは，最下位ビットを1にすることで表現した。m＝30，k＝20，n＝600，席次及び志望順位をランダムとし，6例について処理を行ったところ，サイクル数はほぼ一定で7～9回となった。一方，前節で示した表5，6の例では，サイクル数は，15707回となる。これにより，平均的には最大手間の場合よりはるかに解き易い問題であることがわかる。m，nをいろいろと変化させて計算を行ったところ，サイクル数はm，nの両方に依存し，m又はnの増加に対し，サイクル数は増加することが示された。但し，それぞれの時間計算量への寄与は正確には得られておらず，その評価は今後の課題である。

6．おわりに

本報告では，有限個の要素をもつ二つの集合の要素間の対応づけを一定の制約条件のもとで行う問題を構成し，制約条件に具体的な要件を与えた場合((i)′～(iv)′)の個体とクラスの対応づけのアルゴリズムを提示するとともに，そのアルゴリズムを実行する際の時間計算量に対する検討を行った。現在の計算手段によっては，本報告で述べた多重ポインタ型のデータ構造を用いる方法が，時間計算量の点で有力である。

本報告2節において設定した問題((i)～(iv))は，アルゴリズムを与えた問題((i)′～(iv)′)に比べて問題が一般化されている。優先順位も一つのクラスについて一つであるとは限らず，一つの個体が一つのクラスに対応づけされるとは限らず，また，規則(iv)の与え方は非常に沢山ありうる。しかし，それらの問題構成に対しても，2節で提示したアルゴリズムを部分的に変更することで対処できるものも多い。優先順位が二つ以上与えられる問題の多くは一つの優先順位の問題に帰着するし，2節で紹介したように，(iii)の最大容数を可変とした場合でも，不変とした場合のアルゴリズムを全くかえずに，最大受容数の用語の解釈をかえるのみで対処することができる。という具合である。今後の課題として，制約条件の変更に関する組織的な考察が期待される。