平成 12 年度

学士学位論文

知識獲得を用いたデータベース圧縮における遺伝的アルゴリズムについて

On Genetic Algorithms for Database Compression Using KDD

1010458　　山崎 聖太郎

指導教員　　坂本 明雄

2001 年 2 月 5 日

高知工科大学 情報システム工学科

要 旨

知識獲得を用いたデータベース圧縮における遺伝的アルゴリズムについて

山崎 聖太郎

近年，大規模なデータベースの利用が一般的になるにつれてデータ保持にかかるコストを下げる事が重要な問題になっている．これを解決するための一つの手段として，データベースを圧縮して格納することが有効であると考えられる．データベースを圧縮する方法には様々な手法が報告されている．特に知識獲得を用いた圧縮手法は，圧縮したままデータベースにアクセスでき，他の圧縮手法に比べアクセス速度が高速であることが報告されている．しかし，この圧縮手法では，圧縮に用いる多数のルールを “どの順序で用いて圧縮を行なうか” によって圧縮率が変化するため，圧縮率を向上させるために適切なルール順序を決定する必要がある．そこで本論文では，知識獲得を用いたデータベース圧縮を行なう上で発生する，最適なルール適用順序の決定問題について，遺伝的アルゴリズム（Genetic Algorithm）を用いたルール適用順序決定法を提案する．

キーワード　データベース圧縮，データベースからの知識獲得，遺伝的アルゴリズム

Abstract

On Genetic Algorithms for Database Compression Using KDD

Yamasaki Seitarou

It has been an important problem to lower such cost to data maintenance as use of a large-scale database becomes general in recent years. It is thought effective to compress and store a database as one means for solving this. Various techniques are reported to the method of compressing a database. The compression technique especially using knowledge acquisition can access a database, compressed, and it is reported compared with other compression techniques that access speed is high-speed. However, by this compression technique, in order for the rate of compression to change the rule of a large number used for compression with "or it compresses by using in which order ", in order to raise the rate of compression, it is necessary to determine a suitable rule application order. Then, in this paper, the method determine rule application order for having used genetic algorithm（GA）about the determination problem of the optimum rule application order which occurs when performing database compression which used knowledge acquisition is proposed.

key words database compression, the knowledge acquisition from a database, genetic algorithm

目次

図目次

第 1 章

序論

近年，大規模なデータベースの利用が一般的になるにつれてデータベースの重要度はますます増加し，その規模も大きくなってきている．その結果，データベースにおいては，ここ数年ディスク価格が急激に下落しているとはいえ，データの保持にかかるコストを下げる事が重要な問題になっている．これを解決するための一つの手段として，データベースを圧縮して，格納することが有効であると考えられる．データベースを圧縮することにより，必要な記憶容量を減らすだけでなく，いくつかの副次的な効果も期待できる．まず，データベースにアクセスする際，扱うデータ量が減少するため，ディスク I/O などにかかる時間が減少し，データベース処理時間を減らす事ができる可能性がある．また，バックアップデータの量を減らすこともできるのでバックアップ時間を短縮することが可能になる．更に，ネットワークを介した分散データベース環境においても同じ記憶容量でより多くの複製を作ることができるようになる．データ転送においても，データを圧縮して転送した方が，転送時間が短くなる．このようにデータベースを圧縮することは様々な面において有効な手段であるが，従来の符号化を用いたデータベース圧縮技術をそのままデータベース圧縮に適用すると，データベースにアクセスする場合には，圧縮したデータベース全体を展開しなければならない．そのため，この方法には蓄積容量が小さくなる代わりに，アクセス速度が低下するという欠点がある．また，ブロックごとに従来の符号化圧縮技術を適用する方法でも，問い合わせのために属性ごとのインデックスを作成しなければならず，データ更新時にかかるコストも高い．これに対して，知識獲得手法を用いてルールを抽出し，そのルールをデータと置き換えることによって圧縮を行なう方法では，圧縮率は従来の方法に比べて低下する場合があるが，圧縮したままデータベースにアクセスできるので，アクセス速度は速くなるもの

と考えられる．ただし，この方法は知識獲得手法を用いているため，圧縮対象となるデータベースは，知識獲得が適用できるデータベースとなる．これらのデータベース圧縮方法においては，知識獲得によって得られた多数のルール候補の中からどのような順序でデータベース圧縮に適用していくかによって圧縮率が変化する．したがって，圧縮率を向上させるためには，適切なルール適用順序を調べる必要がある．しかし，多数あるルール候補は互いに複雑に関連しており，単純に全ての組合せを調べて最適な圧縮率になるようなルール適用順序を求めようとすると，ルール候補数を N としたとき，最悪 $O(N!)$ の計算量がかかってしまう．そこで，本研究では，知識獲得を用いたデータベース圧縮を行なう上で発生する，最適なルール適用順序の決定問題について，遺伝的アルゴリズムを用いたルール適用順序決定法を提案する．

第 2 章

知識獲得を用いたデータベース圧縮

2.1 概要

知識獲得を用いたデータベース圧縮方法はデータベースから知識獲得手法を使ってルールを抽出し，その抽出したルールを該当するデータと置き換える事によって，データ量を減らす．抽出したルールの記憶方法には，演繹データベースに格納する方法と関係データベースに格納する方法の 2 種類の方法がある．演繹データベースに格納する方法では，抽出したルールを演繹データベースの IDB（intensional database）に格納し，ルールで記述できなかったデータを演繹データベースの EDB （extensional database）に格納する．一方，関係データベースに格納している場合では，通常の表形式でルールを格納しておくと同時に，抽出したルールからビュー定義を作成して，もとの表をビューとして復元する．ビュー定義によってもとの表が復元されているので，ビューに直接アクセスできる．これらの圧縮方法の利点は，データベースを圧縮しても，圧縮したままデータベースにアクセスできることである．

2.2 圧縮の実行例

図 2.1 のように，ID，A，B，C，D の五つの属性をもつ表 T を圧縮することを考える．そのためにまず，知識獲得アルゴリズムを用いてルールの候補を見つけ出す．ここでは，知識獲得アルゴリズムとして，アプリオリアルゴリズムを用いる．

図 2.1 の表 T に対して，閾値を 2 としてアプリオリアルゴリズムを適用すると図 2.2 のよ

ID	A	B	C	D
1	a	c	e	f
2	a	c	e	f
3	a	c	e	f
4	b	d	e	f
5	b	d	e	f
6	h	k	e	f
7	i	l	e	f
8	j	m	e	f
9	g	n	e	f
10	g	o	p	q

図 2.1　表 T

(-a---)	(-ac--)	(--c-f)	(-bde-)
(-b---)	(-a-e-)	(--de-)	(-bd-f)
(--c--)	(-a--f)	(--d-f)	(-b-ef)
(--d--)	(-bd--)	(---ef)	(--cef)
(---e-)	(-b-e-)	(-ace-)	(--def)
(----f)	(-b--f)	(-ac-f)	(-acef)
(-g---)	(--ce-)	(-a-ef)	(-bdef)

図 2.2　表 T に対するルール候補

うなルール候補が抽出される．例えば，属性 A の要素が a であるタップルは二つ以上あるのでルール候補になる．これを，(-a---) と表す．

抽出されたルール候補の中から，例えば (-acef) を圧縮に適用するとする．まず，(-acef) の要素を含むタップルの中でルールに当てはまらない属性の要素（この場合は ID の要素）を分割表として別の表に格納する（1 番目のルールということで分割表 1 とした）．そして，分割表 1 の ID 以外の属性値が a，c，e，f であることを表すルールを作成し（1 番目のルールなので，ルール 1 とした），もとの表から (-acef) をもつタップルを削除する．その結果，図 2.3 のようになる．以下同様にして，(-bdef)，(-a---) の順序で圧縮に適用したとする

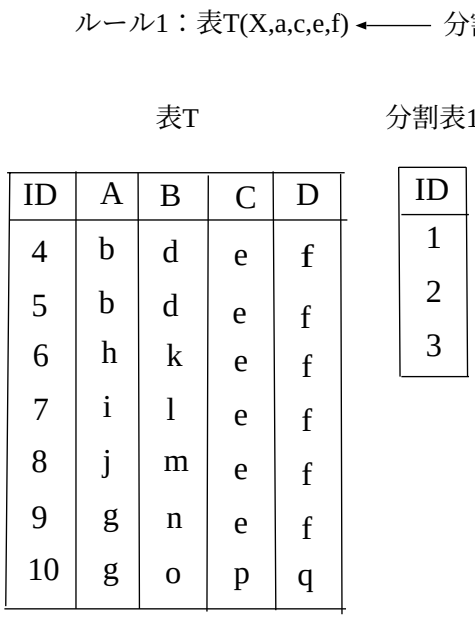

図 2.3　(-acef) を圧縮に適用した結果

と，分割表 2, 3 とルール 2, 3 が作成され，圧縮結果は図 2.4 のようになる．

圧縮の結果，圧縮前に 50 個あった要素数が 22 個になり，要素数はもとの 44 パーセントになったことになる．

2.3　ルール選択

2.2 で示した例では，単純に減った要素数を数えて圧縮率を計測していたが，実際はルールを記憶しておくためのデータ量や，分割表のヘッダサイズなどの増加するデータ量が存在する．また，それらの増加するデータ量を考慮すると，ルール候補から圧縮に適用する順序によって圧縮率が変化する．したがって，圧縮率を向上させるためには，適切なルール適用順序を決定する必要がある．そこで，まずルール候補を圧縮に適用することによって，実際に減少するデータ量を減少量と呼び，次の式で計算する．

$$rd \times rt - RC(rd, rt)$$

ただし，ルールのデータサイズを rd，ルールになるタップル数を rt，ルールを作成することで増加するデータ量を RC とする．増加するデータ量 RC は，ルールの記憶方法や使

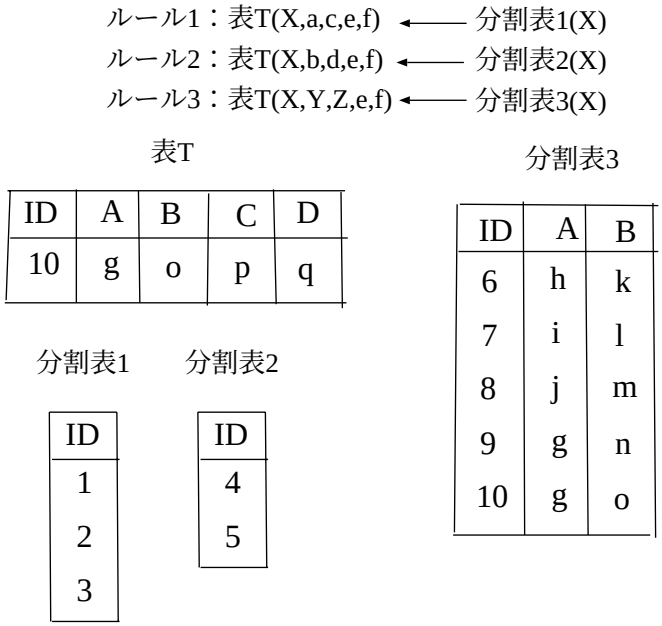

図 2.4　表 T の圧縮結果

(---ef)　(-ace-)　(-ac-f)
(-a-ef)　(--cef)　(-acef)
(-bdef)

図 2.5　減少量が正数である表 T に対するルール候補

用するデータベースによって変化するが，ルールのデータサイズ rd と，ルールになるタップル数 rt の関数として表すことができる．次に，簡単なルール選択方法を用いて図 2.1 の表 T を圧縮し，圧縮率が変化することを示す．簡単のため，要素一つのデータサイズを 1，$RC(rd, rt) = rd + rt + 1$ とする．例えば，(-acef) の減少量は $4 \times 3 - (4 + 3 + 1) = 4$ と計算される．また，(-g---) の減少量は $1 \times 2 - (1 + 2 + 1) = -2$ と負になり，データ量が逆に増えてしまうので，減少量が正数であるものをルール候補としている．その結果，表 T のルール候補は図 2.5 のようになる．

(1) ルールになる要素数の多いルール候補から圧縮に適用した場合　この場合，まず要素数 4 のルール候補 (-acef) と (-bdef) が圧縮に適用される．(-acef) と (-bdef) を圧縮に適用すると，(---ef) のルールになるタップル数 rt は 4 に減少し，(-ace-)，(-ac-f)，

(-a-ef)，(--cef) はルールになるタップル数 rt が 0 になってしまうので，ルール候補ではなくなる．このため，残っているルール候補の中で要素数の多い，(---ef) が圧縮に適用されることになり，減少量の合計は次のように計算されている．

$$4 \times 3 - (4 + 3 + 1) + 4 \times 2 - (4 + 2 + 1) + 2 \times 4 - (2 + 4 + 1) = 6$$

(2) 減少量の値が最大のルール候補から圧縮に適用した場合　すべてのルール候補群について減少量の値を計算すると，(---ef) が最大になる．(---ef) を圧縮に適用すると，他のルール候補はルールになるタップル数が 0 になり，ルール候補でなくなるため，減少量の合計は次のようになっている．

$$2 \times 9 - (2 + 9 + 1) = 6$$

(3) 最適なルール適用順序で圧縮に適用した場合　(-acef) を圧縮に適用すると，(---ef) のルールになるタップル数 rt は 6 に減少するので，減少量の合計は次のように計算される．

$$4 \times 3 - (4 + 3 + 1) + 2 \times 6 - (2 + 6 + 1) = 7$$

このとき，減少量の合計が最大になり，圧縮率が最適になる．

このように，ルール選択によって減少量の合計が変化し，圧縮率も変化してしまう．また，簡単なルール選択方法では，最適なルール適用順序にならない場合が存在する．ルールを最適化する上で，一番困難な問題点はあるルール候補を圧縮に適用すると，他の関連するルール候補のルールになるデータ領域が変化する点である．例えば，(-acef) を圧縮に適用すると，(---ef) のルールになるタップル数が 3 減少し，(-ace-)，(-ac-f)，(-a-ef)，(--cef) はルールになるタップル数が 0 になってしまうので，ルール候補ではなくなる．このように，ルール候補を一つ圧縮に適用すると今までの状態が変化してしまうので，最適なルール選択をするには変化した後の状態まで考慮しなければならない．

第 3 章

遺伝的アルゴリズム

3.1 組合せ最適化問題

遺伝的アルゴリズムは，生物進化の過程を模倣したアルゴリズムであり，選択淘汰，交叉，突然変異といった遺伝的操作を繰り返しながら，世代を重ねることによってより環境への適応度の高い個体を生み出す最適化手法である．

組合せ最適化問題とは，組合せ，離散的な制約条件のもとで，ある目的関数を最小化（あるいは最大化）する数理計画問題であり，一般に次のように記述される [1].

$$min_x f(x) \tag{3.1}$$

$$subject\ \ to\ \ x \in F \tag{3.2}$$

$$(F) \subseteq X \tag{3.3}$$

ここで基本的空間 X およびその部分集合である可能領域（解空間）F は組合せ論的，離散的なものである．組合せ最適化では，実数集合を対象とした連続性や微分の概念に基づく古典的な最適化手法を直接利用することはできない．したがって，組合せ最適化問題の解法は連続変数の最適化手法と本質的に異なるものとなり，一般に解を調べあげるというアプローチにならざるを得ない．しかし，解空間 F の要素（組合せ）の総数は有限ではあっても膨大な数にのぼることが多く，実際に列挙，探索する範囲をいかに限定するかが重要となる．ここで，最適解あるいは準最適解を探索するという観点から組合せ最適化手法を捉えると，その基本的な考え方（アプローチ）は，図 3.1 に示すように，

(a) 列挙的手法　すべての解を列挙して（あるいは，これと等価な手続きによって）厳密な最適解を見出す.

(b) 発見的手法　最適解あるいは準最適解を生成するためのアルゴリズムまたは，ルールを利用して，ただ一つの解を求める.

(c) 探索的手法　列挙的手法と発見的手法の中間に位置する方法で，解空間の部分空間内を探索することによって，（準）最適解を探し出す.

の三種類に分類される．ここで取り上げる遺伝的アルゴリズムは，探索的手法の範疇に入る方法である．この方法では

(A) 良い解を見つけるためには　できるだけ広い領域を探索すること (diversification, exploitation)

(B) 計算時間を短縮させるためには　探索履歴などを利用してできるだけ探索領域を狭めること (intensification, exploitation)

といった相反する要求をうまくバランスさせ， 解の精度向上と計算時間の高バランス化を図ることが肝要である.

3.2　遺伝的アルゴリズムによる組合せ最適化

遺伝的アルゴリズム（GA）による最適化の枠組を示す．GA では，まず問題の解候補を記号列からなる個体（遺伝子列=染色体）に対応させる．言い換えれば，実際の解候補が表現型（phenotype）であり，記号列の個体が遺伝子型（genotype）である．そして，個体に対して，選択（selection）あるいは複製（reproduction）, 交叉（crossover）および突然変異（mutation）などの演算（遺伝的操作：genetic operators）を繰り返し適用することによって，適応度の高い個体（すなわち質の良い解候補）を逐次生成していくことになる．ここで単一の個体を扱うのではなく個体の集合（個体群：population）を対象として，その適応度を高めていくこと，個体を表す記号列そのものだけでなくその部分列（building block）も評

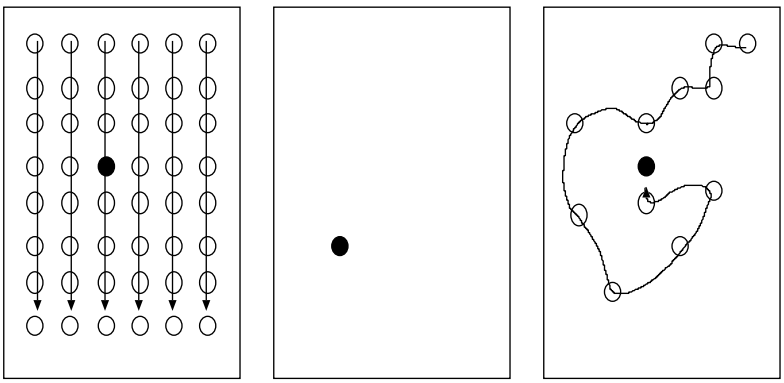

図 3.1　組合せ最適化手法を捉える基本的な考え方（アプローチ）

価され，個体の適応度を高めるのに役立つ部分列が選択によって増えていくこと（implicit parallelism）が特徴である．これらの特徴により解の精度向上と計算時間の高バランス化を図ることが期待される．

3.3　遺伝的アルゴリズムの構成方法

一般に，最適化問題に対して遺伝的アルゴリズム（GA）を構成する場合の手順は以下のようになる [1].

対象となる問題のモデル化，定式化については,

(1) 決定変数および制約条件の選定：問題の解空間 F の設定

(2) 評価値の定量化：最小化あるいは最大化すべき評価関数（目的関数）f の設定

をしなければならない．次に，GA の適用方法について,

(3) 問題の解候補を遺伝子表現（コーディング）する方法の決定：GA による探索の対象となる空間（探索空間）S の設定,

(4) 表現型への変換（デコーディング）方法の決定

(5) 適応度の定量化：目的関数 f と適応度関数 g との対応規則を定める

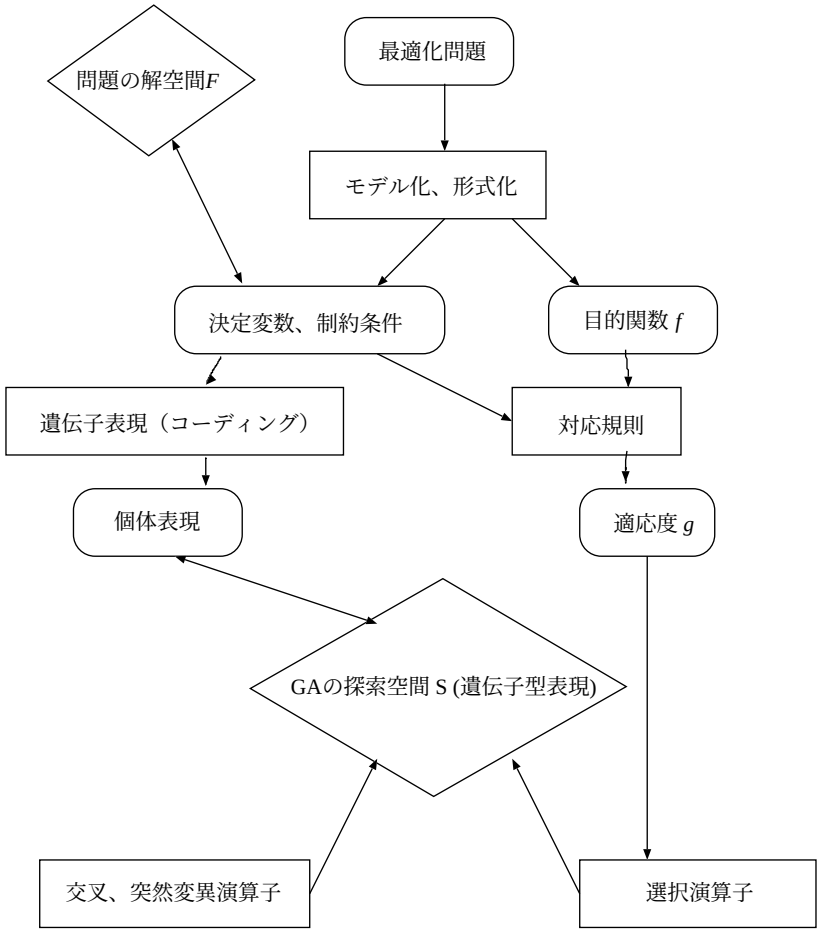

図 3.2　GA 構成の手順

(6) 遺伝演算子の具体化：交叉，突然変異および選択演算の方法を定める

を行なわなければならない．ここでの GA 構成の手順をまとめると，図 3.2 のようになる．

3.4　制約条件の取扱い方法

GA による探索空間についてみると，問題の制約条件 (3.2) 式および (3.3) 式を満たす可能領域（解空間）全体を効率的に探索することが要求される．実際の工学的最適化においては，この（一般に複雑な）制約条件を満足させることが極めて重要であり，GA を構成する場合においても，制約条件をどのような形で取り扱うのかを決めることが肝要である．

3.4 制約条件の取扱い方法

(1) GA による探索空間 S と問題の解空間 F との対応が “1 対 1” になるように探索空間 S を定める，という方法がある．この際には，交叉や突然変異演算によって生成される新しい候補解も制約条件を満たす実行可能解となるようにしなければならない．

これを実現する方法としては，

(1-1) 常に解空間 F に含まれる候補が生成されるように交叉演算子および突然変異演算子を定める．

(1-2) 解空間 F に含まれる候補が生成されるまで交叉や突然変異を繰り返す．

などが考えられる．(1-1) は，場合によって交叉，突然変異演算子の構成にかなりの工夫を要し，(1-2) は計算効率の低下を覚悟しなければならない．方法 (1) または (1-1) の実現が難しい場合の次の方法として，解空間 F を含むような領域に対応するように探索空間 S を設定しておき，

(2) 探索空間 S 内の各点を，解空間 F 内の点（実行可能解）に “変換” する手続きを組み込む，という方法が考えられる．すなわち，探索空間と解空間の関係を多対 1 の関係に緩めるのである．この場合，交叉演算子および突然変異演算子の設計にかなりの自由度が得られる．一方，探索空間が解空間に比べて広くなるので，一般には探索効率の低下が予想される．

次に，制約条件を満足させる解の生成手続きを与えることも困難な場合には，

(3) 解空間 F に含まれない候補にはペナルティを課す，という方法がある．この場合，GA によって得られる解が実行可能解となることは一般には保証されず，ペナルティが少ない場合には，探索が制約を侵す解に収束することも起こり得る．このため，ペナルティの量を調整するなどの作業が必要になってくる．

ペナルティを与える方法の極端な場合として，制約条件を満たさない解候補の目的関数値を無限大（最大化問題の場合は無限小）とし，受理されることを禁止する方法もある．この

3.4　制約条件の取扱い方法

方法は，制約条件を侵す解を生成しないという利点を持つ反面，制約条件が厳しい場合には，交叉や突然変異によって新しい探索空間内の点を生成することが困難になりやすくなる．

第 4 章

知識獲得を用いたデータベース圧縮における遺伝的アルゴリズムの構成方法

知識獲得を用いたデータベース圧縮における遺伝的アルゴリズムの構成方法について述べる．

4.1 問題のモデル化，定式化

知識獲得を用いたデータベース圧縮方法においての利点は，圧縮したデータ量のままのデータベースを扱うことができ，このことによって，データ保持のコストの低下の実現，副次的な効果としてデータ量の減少に伴って，データベースの複製，バックアップ管理にかかるコストを低下させ，また，場合によってはアクセス速度の向上も期待できるようになることだった．その問題点は抽出した多くのルール候補を圧縮に適用する際，ルールの圧縮適用順序によって圧縮率に変化があらわれることである．これは，それぞれのルールには圧縮に適用できるデータ領域が存在していて，それらのデータ領域はほとんどの場合他のルールのデータ領域と重複しているためである．あるルールが適用されたデータ領域は他のルールでの圧縮ができなくなり，したがって，一つのルールが圧縮に適用される毎に重複するデータ領域を持つ他のルールの圧縮可能なデータ領域は減少，または 0 に変化してしまう．このため最適なルール適用順序を決定するには，変化した後の状態まで考慮しなければならない．

No.1	(-a---)	No.8	(-ac--)	No.15	(--c-f)	No.22	(-bde-)
No.2	(-b---)	No.9	(-a-e-)	No.16	(--de-)	No.23	(-bd-f)
No.3	(--c--)	No.10	(-a--f)	No.17	(--d-f)	No.24	(-b-ef)
No.4	(--d--)	No.11	(-bd--)	No.18	(---ef)	No.25	(--cef)
No.5	(---e-)	No.12	(-b-e-)	No.19	(-ace-)	No.26	(--def)
No.6	(----f)	No.13	(-b--f)	No.20	(-ac-f)	No.27	(-acef)
No.7	(-g---)	No.14	(--ce-)	No.21	(-a-ef)	No.28	(-bdef)

図 4.1　抽出したルールに ID 番号を付与

染色体

27 18 21 15 12 28 2 6 4 23 8 20 17 26 13 22 25 11 3 16 24 19 10 9 7 1 5 14

ルールID27(-acef), ID18(---ef) の順序で
圧縮に適用して,ルール候補がなくなって圧縮終了

染色体

18 27 21 15 12 28 2 6 4 23 8 20 17 26 13 22 25 11 3 16 24 19 10 9 7 1 5 14

ルールID18(---ef)を
圧縮に適用して,ルール候補がなくなって圧縮終了

図 4.2　ID 番号の順列による個体の遺伝子型表現

4.2　GA の適用方法

この問題について遺伝的アルゴリズムを用いた解法を構成する．問題の解候補はルールの適用順序であるから，抽出したルールにそれぞれ ID 番号を与え，その ID 番号の順列を，圧縮開始から終了までのルール適用順序として個体の遺伝子型表現に用いる (図 4.1 と図 4.2 を参照)

それぞれのルールは 1 度だけの圧縮適応しかできないので，順列は抽出した全ての ID 番号を 1 度だけ使った順列となり，その長さは抽出したルール数分になる．順列の先頭から最後までの ID 番号の順列は，ID 番号の示すルールを順番に圧縮に適用していくことを示す．順列の途中では適用できなくなったルールの ID 番号は無視して，順列の最後までルールを

圧縮に適応し終えて終了する．これによって，遺伝子型と問題の解候補とは多対 1 に対応することになる．

圧縮目的はデータ量の減少であるが，ルールを適用した際のデータの減少量が負数になり，逆にデータ量が増えてしまうこともある．従って適応度を正数として扱う時，データの減少量は採用できない．そこで，適応度関数には圧縮後のデータベースのデータ量の合計の逆数とする．圧縮後のデータベースのデータ量の合計を以下の式によって求める．

要素一つのデータサイズを 1，ルール ID 番号一つのデータサイズを 1，未圧縮部分のデータベースの要素数を Td，表のタップル数を Tt，圧縮に適用したルールの圧縮順番号を x，最後に適応したルールの順番号を M_x，$R(x)d$ は適用順序 x 番目のルールの要素数を $R(x)d$,$R(x)t$ は適用順序 x 番目のルールの分割表のタップル数を $R(x)t$，圧縮後のデータベースのデータ量を $C-data$ とすると，

$$C-data=\sum_{x=1}^{M_x}(R(x)d+R(x)t)+Td\times Tt \tag{4.1}$$

適応度を f とすると，

$$f=\frac{1}{C-data} \tag{4.2}$$

4.3　交叉演算子

続いて順列の順序の保存のため，交叉演算子には順序交叉 (order crossover) を用いる．まず，乱数を用いてランダムな確率で決定した 2 つの交叉点の値を準備する．2 つの交叉点の値が等しいときはもう一度乱数から新しい 2 つの交叉点の値を得る．乱数を用いて 2 つの交叉点を決定した後，図 4.3 のように交叉対象となっている 2 点が 3 と 6 の場合は，順列のルール ID の 3 点から 6 点までの部分順列を交叉相手の部分順列とそのまま交換する．そして，残りの順列 1 点のルール，2 点，7 点，8 点，9 点のルール ID の配置は交叉前の順列表現により，順序関係を復元する．

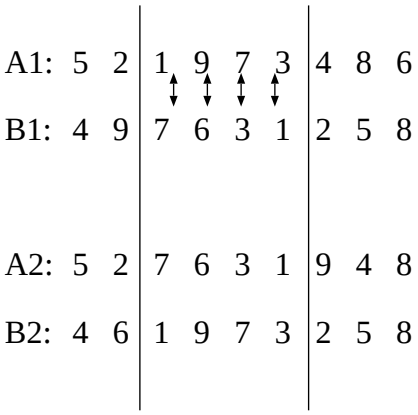

図 4.3　順序交叉 (order crossover) の例

図 4.3 の場合には，A2 の順列の配置で復元するルール ID は 4 と 9 と 2 と 5 と 8 である．A1 においてのこれらのルール ID の順序関係を用いて A2 を復元する．

1. A 1 の 1 点にはルール ID 5 がある．これは復元するルール ID なので A2 の 1 点にルール ID 5 を復元する．
2. A 1 の 2 点にはルール ID 2 がある．これは復元するルール ID なので A2 の 2 点にルール ID 2 を復元する．
3. A 1 の 3 点にはルール ID 1 がある．これは復元するルール ID ではない．すでに A2 に存在する．
4. A 1 の 4 点にはルール ID 9 がある．これは復元するルール ID なので A2 の 7 点にルール ID 9 を復元する．
5. A 1 の 5 点にはルール ID 7 がある．これは復元するルール ID ではない．すでに A2 に存在する．
6. A 1 の 6 点にはルール ID 3 がある．これは復元するルール ID ではない．すでに A2 に存在する．
7. A 1 の 7 点にはルール ID 4 がある．これは復元するルール ID なので A2 の 8 点にルール ID 4 を復元する．
8. A 1 の 8 点にはルール ID 8 がある．これは復元するルール ID なので A2 の 9 点に

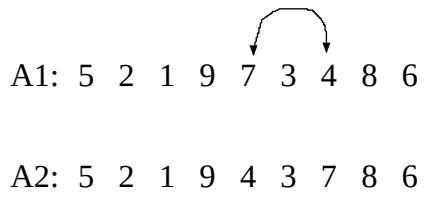

図 4.4　1 点突然変異の例

ルール ID 8 を復元する．

9. A 1 の 9 点にはルール ID 6 がある．これは復元するルール ID ではない．すでに A2 に存在する．

同様に B2 の復元も B1 の順序関係を用いて行なう．

4.4　突然変異演算子

突然変異演算子には，1 点突然変異を用いる．順列表現の 2 点をランダムに選び，その 2 点にあるルール ID を交換する操作である．図 4.4 の場合には 5 点と 7 点のルール ID を交換している．

4.5　選択淘汰

選択淘汰とは，適応度の高い個体集団を得る為に世代ごとに個体集団の取捨選択を計る遺伝的操作である．提案手法ではルーレット戦略を用いて，個体集団全体の適応度の和の中での占有率から選択淘汰を行なう．ここで適応度の高い個体は次世代に多くの子孫，つまり自身の複製を残すことができる．一方，適応度の低い個体は次世代に子孫を残す余地が与えら

個体Aの適応度が0.001,
個体Bの適応度が0.002,
個体Cの適応度が0.0005,
個体Dの適応度が0.0005のとき

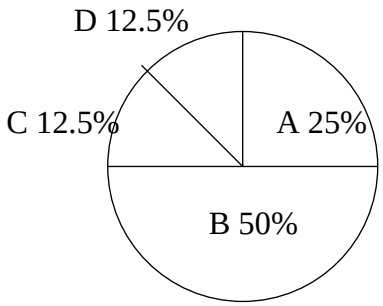

図 4.5　ルーレット戦略を用いた個体の選択淘汰率

れない.

図 4.5 の個体 A の選択淘汰率は

$$0.001/(0.001 + 0.002 + 0.0005 + 0.0005) = 0.25$$

個体 B の選択淘汰率は

$$0.002/(0.001 + 0.002 + 0.0005 + 0.0005) = 0.5$$

個体 C の選択淘汰率は

$$0.0005/(0.001 + 0.002 + 0.0005 + 0.0005) = 0.125$$

個体 D の選択淘汰率は

$$0.0005/(0.001 + 0.002 + 0.0005 + 0.0005) = 0.125$$

となる.

4.6　GA の処理の流れ

全体としての GA の処理の流れは図 4.6 のようになる.

4.6　GA の処理の流れ

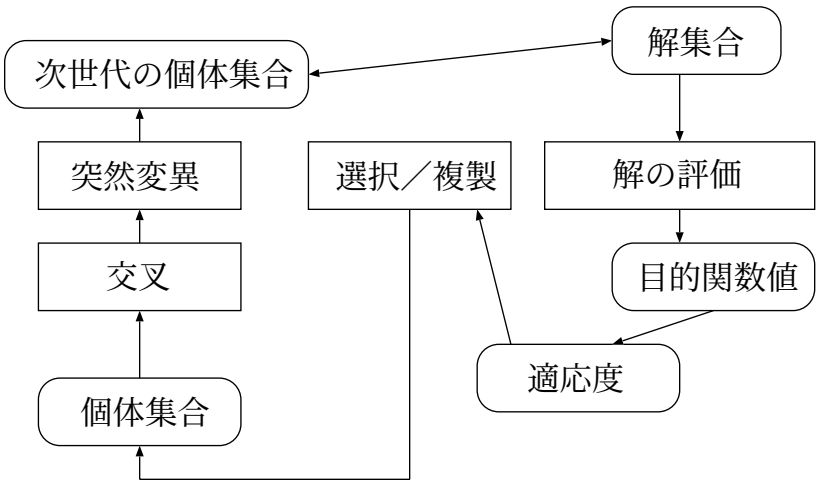

図 4.6　全体としての GA の処理の流れ

試行実験によって最適値を得るパラメータとして初期設定値,

個体集団の個体数 M，交叉確率 p_c，突然変異確率 p_m，終了世代数 T を与え GA の処理が開始する．終了世代数まで処理を行なった時点で終了する．最後に終了世代の個体集団の中から最も適応度の高い個体の表現するルール ID 順列をルール適用順序として得る．

第 5 章

結論

本研究では，知識獲得を用いたデータベース圧縮方法において，遺伝的アルゴリズムを用いたルール適用順序選択法を提案した．提案した方法は任意の圧縮ルール適用順序を順列表現によって表現し，GA による効率的な最適解探索を行なう手法である．

今後の課題としては，GA 設定パラメータの試行実験による最適化を行ない，他アルゴリズムとの比較実験を行なうことが必要である．知識獲得を用いたデータベース圧縮方法は，どのようなルールが抽出できるかによって圧縮率が変化する．したがって，今回はアプリオリアルゴリズムのみを用いたが他の知識獲得アルゴリズムとの組合せにより，更に圧縮できるルールを抽出し，また，GA と他アルゴリズムとの併用，有効な選択淘汰戦略の発見による探索能力の向上が考えられる．

謝辞

本研究をまとめるに当たり，多くのご教示を頂いた坂本 明雄教授，本稿の執筆に当たり適切な助言を頂いた橋本 学氏に感謝いたします．

参考文献

[1] 北野宏明, “遺伝的アルゴリズム 3”, 産業図書, p4–11, 1997.

[2] 相坂一樹, 塚本昌彦, 春本要, 西尾章治郎, “知識獲得を用いたデータベース圧縮のためのルール選択方法について ”, 電子情報通信学会論文誌, Vol.J83-D-I, No.2, pp.273–281, 2000.

[3] Chien-LeGOH, Kazuki AISAKA, Masahiko TSUKAMOTO, Kaname HARUMOTO, Shojiro NISHIO, “Database Compression with Data Mining Methods”, Department of Information Systems Engineering Graduate School of Engineering Osaka University, Japan.

[4] Rakesh Agrawal, Ramakrishnan Srikant, “Fast Algorithms for Mining Association Rules”, IBM Almaden Research Center 650 Harry Road, San Jose, CA 95120.