逆導出原理と遺伝的アルゴリズムを用いた規則集合獲得手法 GA-CIGOL

山本公洋† 鈴木英明†
内藤昭三† 伊藤正樹†

論理プログラミングの枠組において，有限個の正負の事例が与えられたときに，それらを説明できる規則集合（プログラム）を獲得するための新たな帰納推論手法 GA-CIGOL を提案する．背景知識を持たないときに，事例から規則集合を獲得する有効な方法として，逆導出原理を用いて新述語を生成する方法が提案されているが，生成される概念候補の有効性を判定することが困難であるという問題がある．本稿では，規則集合の選択方法について考察し，規則集合の推定では候補を比較することが重要である，との仮説をたてる．また，仮説に基づき，遺伝的アルゴリズムを用いて規則集合を推定する手法を提案し，計算機実験によりその有効性を検証する．

GA-CIGOL: A Concept Acquisition Method Based on Genetic Algorithm and Inverse Resolution

KIMIHIRO YAMAMOTO,† HIDEAKI SUZUKI,† SHOZO NAITO † and MASAKI ITOH †

This paper proposes an inductive inference method (GA-CIGOL) which can generate rule sets to explain a given set of positive or negative examples in the framework of logic programming. Inverse resolution method was proposed for effectively creating new predicates to automatically acquire a new concept from examples. With this method, however, it is difficult to select a target concept from created candidate concepts. This paper, therefore, investigates how to select a target concept, and hypothesizes that comparison among candidate concepts is an important process in selection. Based on the hypothesis, we propose a new concept acquisition method based on a genetic algorithm and an inverse resolution, and evaluates the performance of the proposed method by some computer experiments.

1. はじめに

本稿では，論理プログラミングの枠組において，正・負事例から，事例を説明できる規則集合（プログラム）を獲得する帰納推論の一手法（GA-CIGOL）を提案し，計算機実験によりその有効性を検証する．

事例から規則集合を獲得する手法は，背景知識として事前に与えられた概念（述語）を用いて規則（論理式）を構成する手法と，発想機構を用いて独自に新概念（理論名辞）を生成し，規則を構成する手法の2つに分類できる[4]．後者の手法のひとつとして逆導出原理が提案されている[8]．しかし，逆導出原理を用いた規則集合獲得では，候補として生成される無意味な述語/論理式を効果的に棄却することが課題である．この課題に対して，従来の逆導出原理に基づく規則集合獲得システム CIGOL[8]では，規則の選択を人間（ユーザ）が行っていた．しかし，求める規則集合の詳細や逆導出原理を用いた規則集合形成過程に精通していなければ規則選択を行えないという問題点が残されていた．GA-CIGOL では，遺伝的アルゴリズムを用いて規則集合候補を選択することにより，この問題点の確率的な解決を試みる．

以下，2章では事例から規則集合を獲得する問題を形式的に述べる．3章では CIGOL における候補選択に関する問題点を考察する．4章では規則集合を規定する特徴について考察し，候補選択に関する仮説をたてる．5章では，遺伝的アルゴリズムにより候補選択の枠組を実現した帰納推論システム GA-CIGOL を提案する．6章では計算機実験により，GA-CIGOL の有効性を示す．

† NTT ソフトウェア研究所
NTT Software Laboratories

2. 事例からの規則集合獲得

規則（論理式）を記述するための言語を$\mathcal{L}$，事例を記述するための言語を$\mathcal{L}_0$($\square\in\mathcal{L}_0\subset\mathcal{L}$)とする．この時，本稿で扱う規則集合を獲得する問題は，以下のように定式化できる[4]．

言語$\mathcal{L}$が与えられた時，$T\vdash\mathcal{L}_0^+$かつ$T\nvdash\mathcal{L}_0^-$となる有限論理式集合$T\subset\mathcal{L}$を求める．

ここで，$\mathcal{L}_0^+$および$\mathcal{L}_0^-$は，それぞれ$\mathcal{L}_0$で記述可能な事例のうち，真，偽の真理値を持つ事例の集合である．

オラクル（人間）により与えられる一部の真理値から，$\mathcal{L}$で記述可能な他の，未知なる事例の真理値を導くことはできない(帰納の問題[11])．このとき，未知なる事例の真理値は複数通りの可能性を持つ．したがって，それら事例を説明する論理式集合Tも複数存在する．

一般に，オラクルから有限個の正・負事例しか与えられない場合，事例に対して無矛盾な論理式集合Tは複数存在する．

特に，新述語（理論名辞）を生成し，論理式集合を獲得する問題では，新述語に対する真理値は与えられない．したがって，新述語を生成し，規則集合を獲得する問題では，事例に対して無矛盾な論理式集合Tは多数存在する．

3. CIGOLによる規則集合の獲得

3.1 逆導出原理

本稿では，規則集合候補を生成する方法として，逆導出原理に着目する．

逆導出原理は，導出と呼ばれる論理プログラミングの推論規則を逆向きに辿る操作に基づいた3種類の仮説候補生成原理からなる．3種類の仮説候補生成原理は各々Truncation，Absorption，Intra-Constructionと呼ばれる．

TruncationはPlotkinの最小汎化（least general generalization）[9),10)]を利用して事例の一般化を行う．Absorptionは因果関係を生成する．Intra-Constructionは新たな概念を生成する．

逆導出原理を繰り返し適用することで，正事例からプログラムを形成することができる．逆導出原理(3つのオペレータ）は事例もしくは規則候補に対して適用され，新たな規則候補を生成する．このいくつかの規則候補の選言結合によって規則集合候補（プログラム候補）が構成される．

逆導出原理を用いた規則集合形成過程の例を**図1**に示す．図1において，右側に＊印が付いている事例もしくは規則候補に対して逆導出原理を適用することで，左側に%印が付いた規則候補を生成することができる．図1の例では，reverse述語の規則集合を獲得する途中で，新概念new(concatenateに相当)が発見されている．

```
{正の事例の集合}
reverse([3,k,1], [1,k,3]).
reverse([e,r], [r,e]).
reverse([3,2,4,3], [3,4,2,3]).
……
        ↓
   (途中省略)
        ↓
reverse([], []).
reverse([A], [A]).
reverse([A,B], [B,A]).
reverse([3,A,B], [B,A,3]).
……
reverse([2,3,4,5,6], [6,5,4,3,2]).
reverse([j|A], [B,C,j]):-reverse(A, [B,C]). *
reverse([5|A], [B,C,5]):-reverse(A, [B,C]). *
reverse([5|A], [B,C,D,5]):-reverse(A, [B,C,D]). *

<intra-constructionを適用する>
  ↓ reverse([], []).
    reverse([A], [A]).
    reverse([A,B], [B,A]).
    reverse([3,A,B], [B,A,3]).
    ……
    reverse([2,3,4,5,6], [6,5,4,3,2]).
  % reverse([A|B],C):-reverse(B,D), new(A,C,D).
  % new(j, [A,B,j], [A,B]).
  % new(5, [A,B,5], [A,B]). *
  % new(5, [A,B,C,5], [A,B,C]). *

<absorptionを適用する>
  ↓ reverse([], []).
    reverse([A], [A]).
    reverse([A,B], [B,A]).
    reverse([3,A,B], [B,A,3]).
    ……
    reverse([2,3,4,5,6], [6,5,4,3,2]).
    reverse([A|B],C):-reverse(B,D), new(A,C,D).
    new(j, [A,B,j], [A,B]). *
    new(5, [A,B,5], [A,B]). *
  % new(A, [B|C], [B|D]):-new(A,C,D)

<truncationを適用する>
  ↓ reverse([], []).
    reverse([A], [A]).
    reverse([A,B], [B,A]).
    reverse([A|B],C):-reverse(B,D), new(A,C,D).
  % new(A, [B,C,A], [B,C]).
    new(A, [B|C], [B|D]):-new(A,C,D)
```

図1 逆導出原理
Fig. 1 The Inverse Resolution method.

3.2 CIGOLの問題点

事例集合に対して逆導出原理を適用していく上での戦略（事例もしくは規則候補の組合せ，オペレータの種別・適用手順）は非決定的であり，ひとつの事例集合から無数の規則集合候補が生成可能である．生成可能な候補には，**図2**に示すような事例に対して無矛盾ではあるが無意味な規則集合候補が多数含まれている．

従来の逆導出原理に基づく規則集合獲得システムCIGOLでは，人間（ユーザ）の判断に基づき，無矛盾な候補から規則集合を選択する(CIGOLのアルゴリズムを**図3**に示す)．すなわち，規則候補が生成されるたびに，その有効性の判断を人間に仰いでいた．しかし，この方法では次のような問題点が生じる．

```
rev([], []).
rev([3, 4, 1, 2, 7, 8, 9], [9, 8, 7, 2, 1, 4, 3]).
rev([3, y, 7, i, o], [o, i, 7, y, 3]).
rev([4, 1, 2, 7, 8, 9], [9, 8, 7, 2, 1, 4]).
rev([y, 7, i, o], [o, i, 7, y]).
rev([a, s, d, f, g, h, a, k, l], [l, k, a, h, g, f, d, s, a]).
rev([A|B], [A, C, D, A]):-rev(B, [A, C, D]).
rev([2, A, 3], [3, A, 2]):-rev([A, 5, 6], [6, 5, A]).
rev([A|B], C):-new337(A, B, C, D, E), rev(D, E).
new337(A, B, [A, C, D, A], B, [A, C, D]).
new337(2, [A, 3], [3, A, 2], [A, 5, 6], [6, 5, A]).
rev(A, [6, 5, 4]):-
        rev([4, 5, 6], [6, 5, 4]),
        new337(6, B, [6|A], B, [6, 4, 5]).
rev([z, e, r], [r, e, z]).
.....
```

図2 無矛盾な規則集合候補
Fig. 2 A meaningless rule set consistent with the examples.

stepCI1 初期設定：規則集合を空にする.

stepCI2 入力：ユーザから事例をひとつ受け取り，規則集合に加える.

stepCI3 候補生成：CPUtime に上限を設定，規則集合中の規則に対してオペレータを適用，規則候補を可能な限り生成.

stepCI4 無矛盾性検証：規則候補を一つずつ規則集合に加えてみて，その規則集合と負の事例の無矛盾性をチェックする．矛盾する規則候補は削除する.

stepCI5-1 選択 1：ヒューリスティック関数を用いて規則候補を選別，ある一定値以上の候補をユーザに提示する.

stepCI5-2 選択 2：規則集合を形成する上での規則候補の必要性をユーザに問い合わせる.

stepCI6 冗長性削除：ユーザが必要であると判定したら，規則候補を規則集合に加え，冗長となる規則を削除する.

stepCI7 終了判定：ユーザの判断に基づき終了判定を行なう．不十分な場合，stepCI2 へ戻る.

図3 CIGOL アルゴリズム
Fig. 3 The CIGOL algorithm.

問題点 1 求めたい規則集合に関する具体的かつ詳細なイメージを人間が事前に把握する必要がある.

問題点 2 人間は，逆導出原理を用いた概念形成に関する高度な判断を，多数回にわたり強いられる.

事例からの規則集合獲得では，あらかじめ規則集合に関するイメージを持っていると仮定することは，帰納推論の問題設定に反する.

4. 規則集合候補の相対性

同じ事例を説明できるプログラムの中でも，多くの人間の意見が反映され，長い年月の間に洗練されてきたものをベンチマークとする[13]（図4に考察対象としたプログラム例を示す）.無矛盾な候補から規則集合を選択する方法に関する示唆を得るために，ベンチマークを対象として，真理値以外で規則集合を特徴付けるメタな情報（バイアス）について考察する.

```
member(X, [X|Ys]).
member(X, [Y|Ys]):-member(X, Ys).
    %%%%% Member関数 %%%%%


append([], Xs, Xs).
append([X|Xs], Ys, [X|Zs]):-append(Xs, Ys, Zs).
    %%%%% Append関数 %%%%%


times(0, X, 0).
times(s(X), Y, Z):-times(X, Y, W), plus(W, Y, Z).
plus(0, X, X).
plus(s(X), Y, s(Z)):-plus(X, Y, Z).
    %%%%% Times（乗算）関数 %%%%%


reverse([], []).
reverse([X|Xs], Zs):-
        reverse(Xs, Ys), append(Ys, [X], Zs).
    %%%%% Reverse関数 %%%%%


insert_sort([], []).
insert_sort([X|Xs], Ys):-
        insert_sort(Xs, Zs), insert(X, Zs, Ys).
insert(X, [], [X]).
insert(X, [Y|Ys], [Y|Zs]):-X > Y, insert(X, Ys, Zs).
insert(X, [Y|Ys], [X, Y|Ys]):-X =< Y.
    %%%%% Insert-Sort関数 %%%%%


quick_sort([], []).
quick_sort([X|Xs], Ys):-
        partition(Xs, X, Littles, Bigs),
        quick_sort(Littles, Ls), quick_sort(Bigs, Bs),
        append(Ls, [X|Bs], Ys).
partition([], Y, [], []).
partition([X|Xs], Y, [X|Ls], Bs):-
        X =< Y, partiton(Xs, Y, Ls, Bs).
partition([X|Xs], Y, Ls, [X|Bs]):-
        X > Y, partition(Xs, Y, Ls, Bs).
    %%%%% Quick-Sort関数 %%%%%
```

図4 ベンチマーク
Fig. 4 Sample programs for survey.

ベンチマークに共通する絶対的(候補が単独で持つ)特徴を調べた．その結果，**冗長性がない**という特徴を観測することができたが，それ以外に，ベンチマークに共通する特徴は見出せなかった．しかし，冗長性に関する特徴だけでは無矛盾な候補の中からベンチマークだけを区別，選択できない.

一方，人間も任意の候補を単独で見ただけでは規則集合としてふさわしいか否かを判断することは困難である（3.2節の問題点 1）.

以上の事実より，本稿では，相対的に有効性の高い規則集合候補を選択することにする．最適化手法を用いて帰納推論の自動化可能性を考察する．最適化手法として遺伝的アルゴリズムを用いる.

5. GA-CIGOL

5.1 概　　要

GA-CIGOL では，候補生成，事例に対する完全性/無矛盾性の検証，比較選択を反復することにより，規則集合を獲得する.逆導出原理を用いて候補生成を行う.また，正負の事例の有限集合に対する完全性/無矛盾性

を検証する．さらに，冗長性削除，比較を行い，規則集合を選択する．冗長性削除には，Reduction Algorithm[1]を用いる．

比較の枠組はSimple GA[2]と呼ばれる遺伝的アルゴリズムで実現する．バイアスは，スカラ値を返す評価関数形式で加える．

生成可能な規則集合候補は無数に存在する．そのすべての候補を比較するのは不可能である．遺伝的アルゴリズムでは，候補を確率的に保持することでこの問題を疑似的に解決する．つまり，候補空間からランダム（試行錯誤的）に候補を選択し比較を繰り返す．収束判定は一般に不可能なので（2章参照），規則集合獲得の終了条件にはあらかじめ定めた反復回数で停止させる方法を用いる．

5.2 アルゴリズム

GA-CIGOLでは，事例集合に対する逆導出原理の適用戦略を染色体にコーディングする．

交叉と突然変異の2種類の遺伝操作を用いて適用戦略（染色体）を組み換えることにより，試行錯誤的に規則集合候補を生成する．交叉では，2つの規則集合候補間で部分的に規則候補を交換する．これにより，重要な規則候補の組合せを作ることができる．また突然変異では，新規に規則候補を生成し，新たな可能性を模索する．

規則集合候補は集団で保持し，各規則集合候補には評価値に比例した生存確率を与える．そして，生き残った規則集合候補に交叉の遺伝操作を施す．これにより，集団全体に重要な規則候補，もしくは規則候補の組合せが広がり，自然淘汰的に規則集合を構成する．**図5**にGA-CIGOLの獲得アルゴリズムを示す．

5.3 コーディング手法

染色体は，遺伝子の列で記述する．**図6**に染色体の例を示す．

各遺伝子は，規則集合候補生成過程の1ステップを表す．ひとつの遺伝子は3個以上の要素を持つリストで表現され，第一要素は適用する逆導出原理オペレータの種類，第二要素以降はオペレータが入力として受け取る事例もしくは規則候補に対応する通し番号を表す．各オペレータが入力として受け取る事例もしくは規則候補の数は，Truncationでは2個，Absorptionでは2個，Intra-Constructionでは2個以上である．

各オペレータは事例もしくは規則候補を入力として受け取り，規則候補を出力する．あるオペレータにより出力される規則候補は，ほかのオペレータの入力となることができる．染色体内部におけるオペレータの入出力関係は，染色体内部で一意な通し番号を規則候

stepGA1 入力：事例を集合で受け取る．

stepGA2 初期設定：事例に対する逆導出原理の適用戦略（染色体）をランダムに複数個作成，各適用戦略に基づき規則集合候補を生成する．

stepGA3-1 比較：規則集合候補を評価関数を用いて比較，各々に対して生存確率を決定．

stepGA3-2 選択淘汰：生存確率に基づき規則集合候補をランダムに選択する．

stepGA4-1 候補生成：選択した規則集合候補が持つ適用戦略に対し交叉の遺伝操作を施し，新しい適用戦略を作成する．

stepGA4-2 候補生成：新しく作成した適用戦略に対し，さらに突然変異の遺伝操作を施し，新しい適応戦略を作成する．

stepGA5 冗長性削除&無矛盾性検証：新しく作成した各適用戦略に基づき規則集合候補を生成する．この際，冗長性を削除する．また，負事例に対する無矛盾性をチェックする．矛盾が生じた規則集合候補の適用戦略は，致死遺伝子とし，元の適用戦略を継承する．

stepGA6 世代交代：候補集団中で，元となった規則集合候補を，新しい規則集合候補へと置き換える．

stepGA7 終了判定：世代交代数をチェックする．世代交代数が一定値に達していない場合，stepGA3-1へ戻る．

図5 GA-CIGOLアルゴリズム
Fig. 5 The GA-CIGOL algorithm.

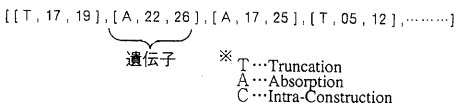


図6 染色体（逆導出原理の適用戦略）
Fig. 6 An operational plan of the inverse resolution method described in the chromosome.

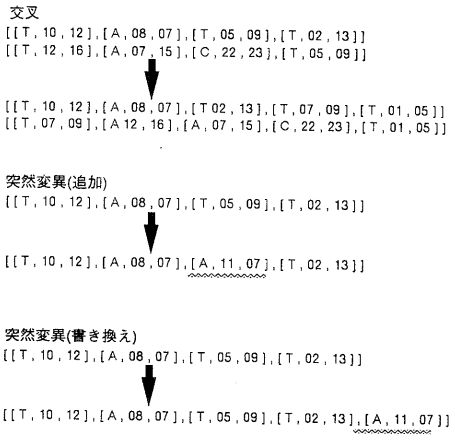


図7 遺伝操作
Fig. 7 Genetic operations for constructing a rule set.

補に与えて統一管理する．

染色体長は可変とする．ただし，無限に長くならないように最大値を設定する．1個の染色体により生成される規則集合候補は一意に定まる．

5.4 遺伝操作

遺伝操作は，交叉と突然変異の2種類を使用する．

交叉では，元の染色体での順序を保存しながら，2個の染色体間で遺伝子を交換する(**図7**参照)．遺伝子交換は，ランダムに行う．すなわち，切断箇所数をランダムに決定し，交叉のためのテンプレートは用いない．

突然変異では，遺伝子の書き換え，もしくは染色体に対して新たな遺伝子の追加を行う（図7参照)．

上記で定めた交叉と突然変異では，致死遺伝子（交配されえない遺伝子）が発生する可能性がある．ひとつの染色体上で，あるオペレータにより出力される規則候補が他のオペレータの入力となる場合がある．遺伝操作によりこの2つの遺伝子間の入出力関係が破壊された場合，入力となる規則候補を失った遺伝子は，致死遺伝子となる．また，染色体長が最大値を越えた場合も致死遺伝子とする．

交叉と突然変異の後で，致死遺伝子を排除するために，遺伝子の入力となる規則候補の有無を検証する．致死遺伝子の検証は，遺伝子の順序が保存されること，オペレータの入出力関係が通し番号を用いて統一管理されることにより，$O(n)$の手間で行うことができる．ここでnは染色体長である．

致死遺伝子が生じた場合には，元の染色体をそのまま次世代に継承する．

5.5 バイアス

本稿では，プログラムの簡潔性に基づく標準的な2種類の評価関数を用いた．

評価関数1は，規則集合の規則数（選言的記述数）Nを値として返す．$1/N$の値を，遺伝的アルゴリズムにおける規則集合候補生存確率とした．

評価関数2は，**図8**の式[8]を用いて規則集合の記述長を計算する．規則集合の記述長Pは，各規則の記述長H_iの総和に1を加える．規則の記述長H_iは，構成要素である各リテラルの記述長t_jの総和に1を加える．リテラルの記述長t_jは，その引数となる項の記述長t_kの総和に2を加える．項の記述長t_jは，その引数となる項の記述長t_kの総和に2を加える．項の一種でもある変数の記述長は1とする．$1/P$の値を，遺伝的アルゴリズムにおける規則集合候補生存確率とした．

$$
\begin{aligned}
\textit{program size } P &= 1+\sum_{i=1}^{l} H_i \\
\textit{horn size } H_i &= 1+\sum_{j=1}^{m} t_j \\
\textit{literal or term size } t_j &= 2+\sum_{k=1}^{n} t_k \\
\textit{variable size } V &= 1
\end{aligned}
$$

図8 バイアス
Fig. 8 A bias used in the GA-CIGOL system.

評価関数1は，単純に選言的記述の簡潔性のみを考慮している．一方，評価関数2は，最終的に，規則集合の記述長Pを用いているが，途中段階で各規則の記述長が考慮されており，このことにより選言的記述(規則数）と連言的記述（各規則の記述長）のトレードオフ機能がもたらされている．

6. 計算機実験

ベンチマーク(member述語，reverse述語，sort述語)およびarch述語，三平方の定理，Michalskiの列車分類規則[7]（**図9**参照）を対象に，規則集合獲得実験を行った．

事例や背景知識はグランドアトムの形でbatch方式で与える．染色体長の最大値は40とした．交叉率は0.5，突然変異率は0.05とした．個体集団の個体数(Population)は100とした．また，終了条件としては世代交代数(Max generation)を使用し，1000世代で終了させた．淘汰圧（Pressure)[2]を設定し，生存確率分布の偏りを周期的に変化させた．淘汰圧の最小値は0.1，最大値は1.0とし，50世代周期で変化させた．

システムはQuintus Prologを用い，SPARC Station 10上に構築した．実験結果を**図10**に示す．また，獲得された規則集合を**図11**に示す．

実験の結果，member述語は評価関数1と評価関数2でベンチマークと同じ規則集合を獲得した．reverse述語は，評価関数1でベンチマークと等価な規則集合を獲得できなかった．評価関数2では，再帰構造の停止条件が多少異なるが，ベンチマークと意味的に等価な規則集合を獲得した．sort述語は，評価関数1と評価関数2でベンチマークと等価な規則集合を獲得でき

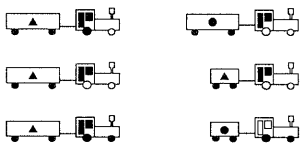


図9 東向き列車と西向き列車
Fig. 9 The two groups of trains.

Target concept	Background knowledge	Number of examples (⊕ / ⊖)	Bias 1 Success/Fail (Generation)	Bias 1 Invented predicate	Bias 2 Success/Fail (Generation)	Bias 2 Invented predicate
member		13 10	Success (57)		Success (12)	
reverse		21 38	Fail (—)		Success (536)	concatenate
sort	≤, >	19 14	Fail (—)		Fail (—)	
arch		11 13	Success (15)		Success (96)	column, brick_or_block
pythagoras	square, plus	14 8	Success (255)		Fail (—)	
train	engine_wheel, car_body, load, car_wheel	27 11	Fail (—)		Fail (—)	

図10 規則集合獲得計算機実験
Fig. 10 The experiments of the rule set acquisition by the GA-CIGOL system.

	評価関数1	評価関数2
[member]	ベンチマークと同じ	ベンチマークと同じ
[reverse]	rev([],[]).rev([A],[A]). rev([A,B],[B,A]). … rev([A,B,C,D,E,F,G],[G,F,E,D,C,B,A]).	reverse([],[]). … reverse([A,B],[B,A]). reverse([X|Xs],Zs):-reverse(Xs,Ys),new1(X,Zs,Ys). new1(A,[B,C,A],[B,C]). new1(X,[Y|Zs],[Y|Ys]):-new1(X,Zs,Ys).
[sort]	sort([A],[A]). sort([A,B|C],D):-sort([B,A|C],D). sort([A,B,C|D],E):-sort([B,C,A|D],E). 十一部分の事例集合	sort([A],[A]). sort([A,B|C],D):-sort([B,A|C],D). sort([A,B,C|D],E):-sort([B,C,A|D],E). 十一部分の事例集合
[arch]	arch(([],beam,[])). arch(([brick|X],Y,[brick|X])):-arch((X,Y,X)). arch(([block|X],Y,[block|X])):-arch((X,Y,X)).	arch((X,beam,X)):-new2(X). new2([]). new2([X|Xs]):-new3(X),new2(Xs). new3(brick). new3(block).
[pythagoras]	pythagoras(X,Y,Z):- square(X,Xs),square(Y,Ys),square(Z,Zs), plus(Xs,Ys,Zs).	事例集合そのまま
[train]	事例集合そのまま	事例集合そのまま

図11 獲得された規則集合
Fig. 11 The rule sets acquired by the GA-CIGOL system.

なかった.

arch 述語は，評価関数1と評価関数2で意味的に等価な規則集合を獲得した．特に評価関数2では，人間の認識に近い柱（column）や石（brick_or_block）の概念を獲得できた．

三平方の定理は，評価関数1で規則を獲得した．しかし，評価関数2で規則獲得に失敗した．

列車分類規則は，評価関数1と評価関数2で規則獲得に失敗した．

7. 考　　察

評価関数1では reverse 述語に関する規則集合を獲得できなかった．これは reverse 述語に関する規則集合の形成に至る途中において，評価関数1では高く評価されない過程が含まれるためである．評価関数2では三平方の定理に関する規則を獲得できなかった．これは三平方の定理に関する規則の形成に至る途中において，評価関数2では低く評価される過程が含まれるためである．さらに，評価関数1と評価関数2両方で列車分類規則を獲得できなかった．列車分類規則は，正事例にも負事例にも存在する属性の連言的記述である．列車分類規則を形成する途中過程で生成される候補は負事例まで満たす危険性がある．事例に対する無矛盾性検証が優先される一方，評価関数1や評価関数2では種別性（強く正事例に共通する特徴）が評価されない．このため，学習に失敗した．これらの結果は，獲得する規則の種類および獲得の全過程で単一の評価関数を用いる GA-CIGOL の限界を示している．

また，reverse 述語や sort 述語に関するベンチマークは，評価関数1や評価関数2で高く（優良に）評価される．しかし，reverse 述語は評価関数1で規則集合獲得に失敗した．また，sort 述語は評価関数1と評価

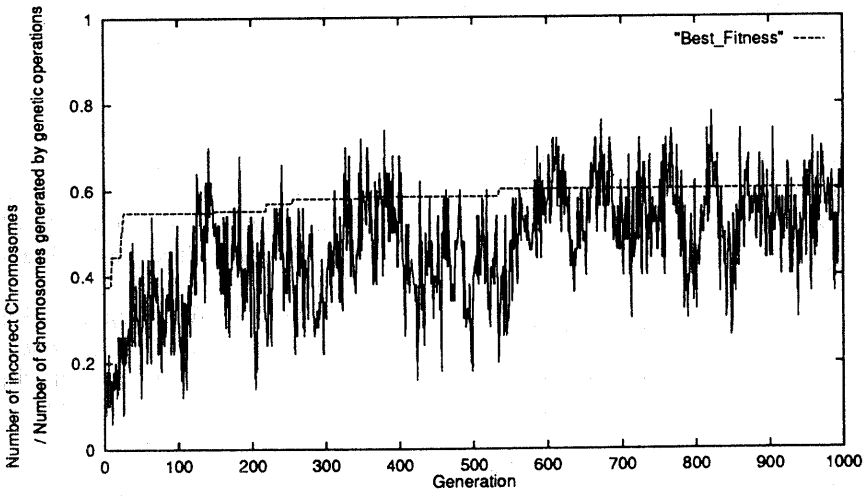


図 12 致死遺伝子発生率
Fig. 12 The generating ratio of incorrect chromosomes.

関数 2 で規則集合獲得に失敗した．さらに，reverse 述語に関しては評価関数 2 で規則集合を獲得できたが，かなりの世代交代数を必要とした．一般に，複雑な規則ほど，規則集合獲得が困難になるのは当然であり，主原因としては，上記の単一評価関数の問題が考えられるが，その他に，コーディング手法の問題が考えられる．今回の実験結果からも示唆されているように，おそらく汎用的な単一評価関数は存在せず，学習対象のタイプと評価関数のタイプとの間には学習可能性あるいは学習効率において密接な相関があるものと考えられる．また，この問題に対しては，多評価関数によるパレート最適性の利用などが有効と考えられるので，今後検討を進めたい．

また一般に，複雑な規則集合ほど，染色体上で遺伝子間に多くの依存関係が存在し，交叉・突然変異の遺伝操作で致死遺伝子となる確率が高くなる．reverse 述語に関する規則集合獲得過程における致死遺伝子発生率の推移を図 12 に示す．同様の問題点が Genetic Programming に関しても報告されている[3),6)]．原因を正確に特定するには，まずコーディング手法の問題を解決する必要がある．

8. 関連研究

従来研究の大半において，規則集合獲得の成功基準は事例に対して無矛盾な規則集合を求めることであった．

代表的な規則集合獲得システムに MIS[12)]がある．MIS は背景知識として事前に与えられる概念から規則を構成する．MIS では，獲得可能な規則集合候補の

```
rev([z, e, r], [r, e, z]).
rev([3, k, l], [l, k, 3]).
rev([2, 4, 3], [3, 4, 2]).
.....
        ↓
rev([A, B, C], [C, B, A]).
```

図 13 特殊な reverse 述語
Fig. 13 A specific kind of the reverse program.

範囲が背景知識の範囲に限定される．また，規則を記述するための，背景知識として与えられる概念の意味（真理値）は事前に与えられる．このため MIS は，比較的少ない事例，もしくはオラクル（ユーザ）に対する質問から規則集合を一意にさだめることができる．

また，オラクルに対して質問することなく，かつ有限個の事例から規則集合を獲得するシステムに CHAMP[5)]がある．CHAMP は背景知識として与えられた概念から規則を構成するシステム CHAM と，新概念を生成するシステム DBC より構成される．CHAM が規則集合獲得に失敗した時，DBC が起動される．CHAMP は，背景知識を用いる CHAM とのハイブリッド型であること，および規則集合形成過程を決定的過程としてとらえアルゴリズム化しているという 2 つの理由により規則集合を高速に獲得する．また，複数種類のバイアスを規則集合形成過程で個別に用いることで複雑な規則集合の獲得に成功している．

しかし，CHAMP では reverse 述語に関する規則集合を獲得するが，3 つの要素より構成されるリストに限定した reverse 述語規則（図 13 参照）は獲得できない．図 13 に示す規則を獲得するために背景知識は必要ない．しかし，CHAM で用いられる精密化オペレー

タ[12]の候補生成能力不足のため，規則候補生成に失敗する．通常の reverse 述語規則獲得が成功するのは，CHAM におけるこの失敗が有効に作用し，DBC が起動されるからである．

これらに対し GA-CIGOL では，獲得可能な規則集合の範囲が限定されることなく，有限個の事例から規則集合を獲得できる．

9. むすび

本稿では，逆導出原理の問題点に関する考察と，ベンチマークに対する調査に基づき，無矛盾な候補から規則集合を選択するには候補を比較することが重要である，との仮説をたてた．

また，仮説に基づき，独自に新概念を生成する規則集合獲得手法として，逆導出原理と遺伝的アルゴリズムを組み合わせた規則集合獲得手法 GA-CIGOL を提案し，計算機実験により獲得能力を検証した．その結果，典型的なプログラムの学習に対しては，提案手法の適用可能性を確認した．

逆導出原理に基づく従来システム（CIGOL）では，規則候補を単独で人間に提示し，選択を行わせていた．このため，求めたい規則集合に関する具体的かつ詳細なイメージを事前に把握する必要があった．

これに対し GA-CIGOL では，仮説に基づき，候補を比較することで規則集合を獲得する．このため，求めたい規則集合の詳細を事前に把握することなく，選択が可能となる．また，人間に委ねられていた規則候補形成に関する高度な判断が不要となり，ユーザの労力負担を軽減できる．

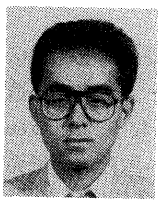
今後は，さらなる計算機実験を行い，提案手法の妥当性を検証するとともに，計算機実験の結果明らかになった単一評価関数，コーディング手法の問題点について考察を行う予定である．

謝辞 日頃より研究へのご支援をいただく NTT ソフトウェア研究所 広域コンピューティング研究部 後藤滋樹部長に感謝します．

参考文献

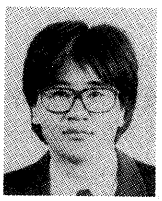
1) Buntine, W.: Generalised Subsumption and Its Application to Induction and Redundancy, *Artif. Intell.*, Vol. 36, No. 2, pp. 149-176 (1988).
2) Goldberg, D. E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison Wesley Publishing (1989).
3) 伊庭斉志：遺伝的プログラミングと進化論的な学習，人工知能学会誌，Vol. 9, No. 4, pp. 512-517 (1994).
4) 川村 正：帰納論理プログラミング—論理プログラミングの帰納的一般化を中心に，コンピュータソフトウェア，Vol. 10, No. 5, pp. 3-15 (1993).
5) ブンサーム・キッスィリクン，沼尾正行，志村正道：弁別に基づく構成的帰納学習，人工知能学会誌，Vol. 7, No. 6, pp. 1027-1037 (1992).
6) Koza, J. R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, The MIT Press, Cambridge (1992).
7) Medin, D. L., Wattenmaker, W. S. and Michalski, R. S.: Constraints in Inductive Learning: An Experimental Study Comparing Human and Machine Performance, *Cognitive Science*, Vol. 11, pp. 299-339 (1987).
8) Muggleton, S. and Buntine, W.: Machine Invention of First-order Predicates by Inverting Resolution, *Proc. of the 5th Int. Workshop on Machine Learning*, pp. 339-352 (1988).
9) Plotkin, G. D.: A Note on Inductive Generalization, *Machine Intelligence 5*, pp. 153-163, Elsevier North-Holland, New York (1970).
10) Plotkin, G. D.: A Further Note on Inductive Generalization, *Machine Intelligence 6*, pp. 101-124, Elsevier North-Holland, New York (1971).
11) Popper, K. R.: *The Logic of Scientific Discovery*, Basic Book, New York (1959).
12) Shapiro, E. Y.: *Algorithmic Program Debugging*, The MIT Press, Cambridge (1983).
13) Sterling, L. and Shapiro, E. Y.: *The Art of Prolog*, The MIT Press, Cambridge (1986).

（平成 6 年 9 月 16 日受付）
（平成 7 年 6 月 12 日採録）



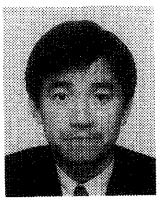
山本 公洋（正会員）

1967 年生．1992 年早稲田大学理工学部電気工学専攻修士修了．同年 NTT 入社．現在 NTT ソフトウェア研究所広域コンピューティング研究部所属．帰納推論，機械学習，要求仕様獲得の研究開発に従事．電子情報通信学会，人工知能学会各会員．



鈴木 英明（正会員）

1961 年生．1985 年早稲田大学理工学部数学科卒業．同年 NTT 入社．現在 NTT ソフトウェア株式会社へ出向中．主に CASE システム，プログラム検索，プログラムの抽象化の研究に従事．日本ソフトウェア科学会，ACM，IEEE 各会員．



内藤 昭三（正会員）

1955年生．1979年京都大学工学部数理工学専攻修士修了．同年NTT入社．現在NTTソフトウェア研究所広域コンピューティング研究部所属．自然言語処理，要求仕様獲得の研究開発に従事．電子情報通信学会，人工知能学会，言語処理学会，計算言語学会，計量国語学会各会員．

伊藤 正樹（正会員）

1953年生．1976年京都大学理学部数学科卒業．1978年同大学院工学研究科数理工学専攻修士課程修了．同年日本電信電話公社（現NTT），武蔵野電気通信研究所入所．以来，ネットワーク構成法，並行システム仕様記述・検証法，要求獲得技術の研究に従事．現在，NTTソフトウェア研究所主幹研究員．工学博士．1987年より1年間，カナダ，モントリオール大学客員教授．IEEE, ACM, Internet Society, 電子情報通信学会各会員．