人　工　知　能 94－1
（1994．6．20）

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[1]　　内藤 昭三　　伊藤 正樹

NTT ソフトウェア研究所
〒180 東京都武蔵野市緑町 3-9-11

概要： 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[1]　Shozo NAITO　Masaki ITOH
kimihiro@slab.ntt.jp　suzuki@slab.ntt.jp　naito@slab.ntt.jp　itoh@slab.ntt.jp

NTT Software Laboratories
9-11 Midori-Cho 3-Chome Musashino-Shi, Tokyo 180 Japan

Abstract: This paper proposes an inductive inference method (GA-CIGOL) which can generate rule sets explaining 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 difficlut 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 proposes 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]現在，NTT ソフトウェアに出向中

1 はじめに

ソフトウェア開発上流工程の要求獲得では，クライアントの要求を正確に把握するために多大な時間と労力を必要とする．また，エキスパートシステム構築においても，エキスパートの体系化された知識を獲得するために，多大な時間と労力を必要とする．計算機を用いて断片的な事例から体系化された規則や有効な新概念を獲得できれば，ソフトウェア開発上流工程の要求獲得に対する計算機支援が可能となり，エキスパートシステム構築における知識獲得ボトルネックと呼ばれる問題の多くを解決することができる．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 [8]．この内，独自に新概念を生成する手法のひとつとして逆導出原理が提案されている [5]．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[5] では，規則の選択を人間（ユーザ）が行なっていた．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を選択することにより，この問題点を解決する．

以下，２章では理論名辞の問題を示す．３章では逆導出原理に基づく帰納推論システム 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．

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，正・負事例から規則集合を獲得する問題は，以下の様に定式化できる．

> 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．

但し，$\mathcal{L}_0^+$（もしくは $\mathcal{L}_0^-$）とは，$\mathcal{L}_0$ で記述可能な事例のうち，真（もしくは偽）の真理値を持つ事例の集合を示す．

言語 $\mathcal{L}$ で記述可能な全ての論理式に対し，部分的にしか真理値が与えられない場合，有限論理式集合 T をユニークに定めることは不可能である．

一般に有限個の正・負事例では，$\mathcal{L}_0$ で記述可能な事例全てに対して真理値を与えることはできない．従って，有限個の正・負事例から T をユニークに定めることはできない．

また，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．従って，新述語を生成し，規則集合を獲得する問題は，事例を無限個与えられたと仮定しても T，もしくは新述語をユニークに定めることはできない．

新述語を生成し，論理式集合を獲得する手法において，無数に存在する述語候補の中で，いつ，どの述語を用いるかを決定するのは難しく，一般に，理論名辞の問題 [3] と呼ばれている．

3 CIGOL による規則集合の獲得

3.1 逆導出原理

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

```
正の事例
    rev([], []).
    rev([3, 4, 5, 6, 7, 8, 9], [9, 8, 7, 6, 5, 4, 3]).
    rev([9], [9]).
    rev([y, u, i, o], [o, i, u, y]).
    rev([2, 3, 4, 5, 6, 7, 8, 9], [9, 8, 7, 6, 5, 4, 3, 2]).
    ..............

    rev([h, j, k, l], [l, k, j, h]).
    rev([l], [l]).
    rev([u, i, o], [o, i, u]).
    rev([5, 6, 7, 8, 9], [9, 8, 7, 6, 5]).


負の事例
    not(rev([1, 2, 3, 4, 5, 6, 7, 8, 9], [1, 2, 3, 4, 5, 6, 7, 8, 9])).
    not(rev([t, u, y, i, o], [o, i, u, y, t])).
    not(rev([s, d, f, g], [l, k, j, h, g, f, d, s])).
    not(rev([h, h, j, k, l], [l, k, j, h, g])).
    ..............

    not(rev([q, w, e, r, t, y, u, i, o], [o, i, u, y, t, r, e, w])).
    not(rev([6, 7, 8, 9], [9, 8, 7, 7])).
    not(rev([y, u, i, o], [o, i, u, y, t])).
```

図 1: 事例

と呼ばれる．

Truncation は Plotkin の最小汎化（least general generalization）[1][2] を利用して事例の一般化を行なう．Absorption は因果関係を生成する．Intra-Construction は新たな概念を生成する．

```
学習経過の例

        正の事例の集合

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 2: 規則集合形成過程

逆導出原理を繰り返し適用することで，図 1に示す様な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組合せ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 2に示す．図 2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 2の例では，reverse 述語の規則集合を獲得する途中で，新概念 new（append に相当）が発見されている．

3.2 CIGOL の問題点

新概念を生成し，規則集合を獲得する問題では，事例を与えても，規則集合をユニークに定めることはできない．また，逆導出原理を事例もしくは規則候補に無作為に適用すると，無意味な述語／論理式を含む規則集合候補（図 3参照）までもが生成される．逆導出原理を用いて規則集合獲得を行なうためには，膨大な候補の中から，何を概念／規則として獲得し，何を概念／規則として獲得すべきでないかを明確にする必要がある．

```
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 3: 無意味な述語／論理式

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，選択を人間（ユーザ）の判断に基づき行なっていた（CIGOL のアルゴリズムを図 4に示す）．しかし，CIGOL における規則集合獲得では，次の様な問題点が生じる．

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

4 規則集合の選択

規則集合を選択する方法を得るために，サンプル・プログラムに共通する特徴・性質を調べた．サンプル・プログラムには，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選んだ (図 5に考察対象としたサンプル・プログラム例を示す)．

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 4: CIGOL のアルゴリズム

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

%% Last関数：
    last(X, [X]).
    last(X, [Y|Ys]):-last(X, Ys).

%% Arch関数：
    arch((X, beam, X)):-column(X).
    column([]).
    column([X|Xs]):-brick_or_block(X), column(Xs).
    brick_or_block(brick).
    brick_or_block(block).

%% Less-Than関数：
    X =< X.
    X =< Y :- successor(X) =< Y.

%% Minimun関数：
    minimun(X, [X]).
    minimun(X, [Y|Ys]):-X =< Y, minimun(X, Ys).
    minimun(Y, [Y|Ys]):-X > Y, minimun(X, Ys).

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

%% Insert-Sort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.
```

図 5: サンプル・プログラム

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，サンプル・プログラムに共通する特徴・性質は見出せなかった．しかし，冗長性に関する特徴だけではサンプル・プログラムと無意味な述語／論理式を区別，選択できない．

また，人間も任意の候補を見ただけでは規則集合としてふさわしいか否か，選択できない（3.2の問題点１）．

以上の議論より，規則集合の選別方法に対して次の様な仮説を設けることができる．

仮説１ 任意の候補が単独で持つ情報のみでは，選択を行なえない

仮説１より，規則集合を選択するには候補間の相対的な関係が必要であると考えられる．いいかえれば，求めたい規則集合は相対的に重要性の高い候補であり，比較を繰り返さなければ認識できないと考えられる．

このことより，選択方法に対して，さらに次の仮説を設けることができる．

仮説２ 規則集合を選択するためには，候補を比較することが重要である．

本稿では仮説２に基づき，比較を行なう概念獲得手法を提案する．

5 GA-CIGOL

5.1 GA-CIGOL の概要

図 6に GA-CIGOL による規則集合獲得の概念図を示す．

候補生成とモデルの同定，選択を行なうことにより，規則集合を獲得する．逆導出原理を用いて候補生成を行なう．また，事例の有限集合を用いて無矛盾性を検証し，モデルを同定する．さらに，冗長性削除，比較を行ない，規則集合を選択する．冗長性削除には，Reduction Algori-thm[4] を用いる．

比較の枠組は遺伝的アルゴリズムで実現する．遺伝的アルゴリズムを用いることにより，単に規則集合候補を比較するだけではなく，複数の比較結果を統合して規則集合を構成する．

逆導出原理を用いた候補生成では，生成可能な候補

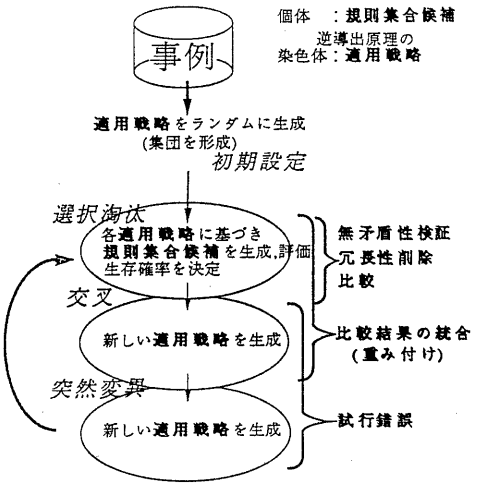

図 6: GA-CIGOL の規則集合獲得概念図

数が組合せ的爆発を起こす．仮説 2 に基づき，全ての生成可能な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，比較を繰り返す．この時，規則集合獲得の終了条件は 2 種類考えられる．

終了条件 1　一定数を定め，有限時間で停止させる

終了条件 2　解の収束性を判定する

提案手法では，終了条件 1 を採用する．

5.2　アルゴリズム

図 7 に GA-CIGOL の獲得アルゴリズムを示す．

GA-CIGOL では，規則集合候補を個体（生物）に見立てる．各個体は 1 本の染色体を持つ．規則集合候補を生成するための，事例集合に対する逆導出原理の適用戦略（オペレータの種別・事例もしくは規則候補の組合せ）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（個体）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，複数の比較結果を統合して自然淘汰的に規則集合を構成する．

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（個体）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（：ルーレット選択）．

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 7: GA-CIGOL のアルゴリズム

5.3　コーディング手法

事例集合に対する逆導出原理の適用戦略を各個体の染色体にコーディングする．

染色体は固定長で，特徴を司る最小構成単位である遺伝子型の部分と情報を持たないスペーサーの部分より構成される．長さが一定になる様に，遺伝子型を適当な長さのスペーサーで区切るものとする．図 8 に染色体の例を示す．

各遺伝子型は，逆導出原理（3 つのオペレータ）の一回の適用（オペレータの種別と適用対象となる事例もしくは規則候補）を表す．ひとつの遺伝子型は 3 つ

の遺伝子座を持ち，第一遺伝子座は適用するオペレータの種類，第二・三遺伝子座は適用対象となる事例もしくは規則候補を通し番号で表す．一回の適用で，オペレータの適用対象となる事例もしくは規則候補の数は2つとする．

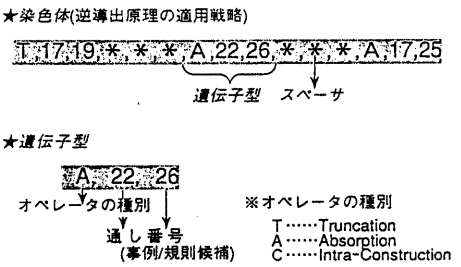

図 8: 染色体

事例および規則候補は，規則候補列を用いて統一管理し，ユニークな通し番号を与える．規則候補列の初期値は正事例とする．新たに生成された規則候補は，その度に規則候補列に追加する．生成された規則候補を規則候補列に追加することで，生成された規則候補に対し，繰り返しオペレータを適用することが可能となる．

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．

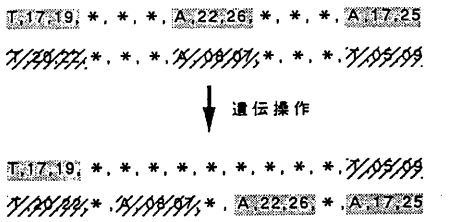

図 9: 交叉

交叉には，遺伝子型単位の複数交叉を用いる（図9参照）．二つの染色体間で任意に遺伝子型を交換する．

一般的な突然変異は染色体を任意に書き換えるが，今回は遺伝子型単位で書き換えを行う（図10参照）．まず，突然変異を生じさせる位置を指定する．その位置がスペーサーの場合（図10場合1），オペレータを任意に選択，可能な組合せの新たな遺伝子型を染色体に追加する．突然変異の位置が遺伝子型上の場合（図10場合2），場合1と同様に新たな遺伝子型を生成，元の遺伝子型と置換する．

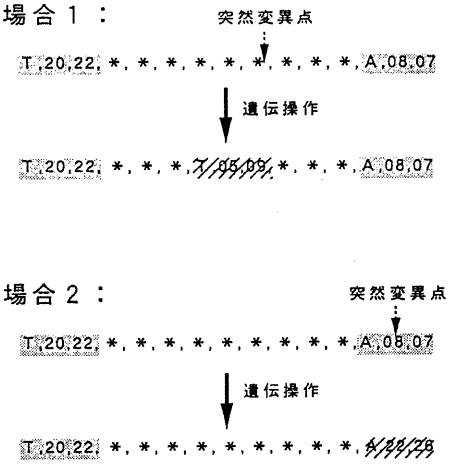

図 10: 突然変異

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，ある遺伝子型により生成される規則候補が，他の遺伝子型が示すオペレータの適用対象となる場合がある．遺伝操作によりこの2つの遺伝子型の関係が破壊された場合，前提条件となる規則候補を失った遺伝子型は，致死遺伝子となる．

交叉と突然変異では，致死遺伝子を回避するために，遺伝子型の前提条件となる規則候補の有無を検証する．

また，遺伝操作に伴い，染色体の長さが一定になる様にスペーサーで調節する．染色体の長さが固定長を越えた場合は，致死遺伝子とし，元の個体の染色体をそのまま継承する．

5.5 規則集合候補の比較選択

規則集合候補の比較は，スカラー値を返す評価関数で試みる．1種類の評価関数（基準）に基づき，候補を比較する．

実験システム作成に際し，2種類の評価関数を用意する．各々，個別に用いて規則集合獲得実験を行なう．

評価関数1は，規則集合の規則数を値として返す．評価関数2は，CIGOLが候補空間を限定するために用いたheuristic関数で，変数，項，リテラル及び節の

総合数（size-of clause set）を値として返す（図11参照）．評価関数1，評価関数2では共に，評価値の低い候補，なるべく記述の単純な候補を規則集合として選択する．

$$\text{size-of clause set}\{C_1,\ldots,C_n\} = 1 + \sum_{i=1}^{n}(\text{size-of clause } C_i),$$

$$\text{size-of clause } \{L_1,\ldots,L_n\} = 1 + \sum_{i=1}^{n}(\text{size-of literal } L_i),$$

$$\text{size-of literal or term } f(t_1,\ldots,t_n) = 2 + \sum_{i=1}^{n}(\text{size-of } t_i),$$

$$\text{size-of variable } V = 1$$

図 11: ヒューリスティック関数

5.6 CIGOLとの対比

CIGOLでは，規則候補を単独で人間に提示し，選択を行なわせていた．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．しかし，これでは規則集合獲得の目的に反する．

これに対しGA-CIGOLでは，仮説2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（自動化）できる．これにより3.2で示したCIGOLの問題点1・2を解決する．

6 実験結果および考察

一部のサンプル・プログラムに対し，規則集合獲得実験を行なった．

個体集団の個体数（Population）は100とした．また，獲得の終了条件として，世代交代数（Max generation）は1000世代とした．さらに，淘汰圧（Pressure）[6]を設定し，生存確率分布の偏りを周期的に変化させた．

獲得された規則集合を図12に示す．また，獲得に要した時間を図13に示す．

実験の結果，member述語は評価関数1と評価関数2でサンプル・プログラムと同じ規則集合を獲得した．arch述語は，評価関数1と評価関数2で獲得した規則集合が異なり，評価関数2でサンプル・プログラムと

獲得された規則集合：

[Member]：評価関数1，評価関数2

```
member(X, [X|Xs]).
member(X, [Y|Ys]):-member(X, Ys).
```

[Arch]：評価関数1

```
arch((X, beam, X)):-new1(X).
new1([]).
new1([brick|Xs]):-new1(Xs).
new1([block|Xs]):-new1(Xs).
```

※ new1 は column に相当する

[Arch]：評価関数2

```
arch((X, beam, X)):-new1(X).
new2(brick).
new2(block).
new1([]).
new1([X|Xs]):-new2(X), new1(Xs).
```

※ new2 は brick_or_block に相当する

[Reverse]：評価関数1

```
rev([], []).
rev([3, 2, 3, 4, 5, 6, 7, 8], [8, 7, 6, 5, 4, 3, 2, 3]).
rev([A], [A]).
rev([A, B], [B, A]).
rev([A, B, C], [C, B, A]).
rev([A, B, C, D], [D, C, B, A]).
rev([A, B, C, D, E, F], [F, E, D, C, B, A]).
rev([A, B, C, D, E, F, G], [G, F, E, D, C, B, A]).
rev(A, [B, C, D, E, F]):-rev([r|A], [B, C, D, E, F, r]).
```

[Reverse]：評価関数2

```
reverse([], []).
reverse([A], [A]).
reverse([A, B], [B, A]).
reverse([X|Xs], Zs):-
  reverse(Xs, Ys), new3(X, Zs, Ys).
new3(A, [B, C, A], [B, C]).
new3(X, [Y|Zs], [Y|Ys]):-new3(X, Zs, Ys).
```

※ new3 は append に相当する

図 12: 獲得された規則集合

同じ結果を得ることができた．reverse述語は，評価関数2で，再帰構造の停止条件が多少異なるが，サンプル・プログラムと意味的に等価な規則集合を獲得することができた．評価関数1では，意味的にもサンプル・プログラムと異なる規則集合しか獲得できなかった．reverse述語は，member述語やarch述語に比べ，獲得にかなりの時間，step数を要した．

Max generation = 1000　Population = 100
Low pressure = 0.1　High pressure = 1.0
Pressure period = 25
Cross-over rate = 0.3　Mutation rate = 0.2

プログラム	正事例数	負事例数	世代交代数	
			評価関数1	評価関数2
member	13	10	57	12
arch	11	13	15	96
reverse	21	38	(267)	536

図 13: 実験結果

実験により，提案手法に関して以下の問題点が明らかになった．

問題点i 獲得する規則集合の種類によって評価関数の設定を変更する必要がある

問題点ii 複雑な規則集合ほど，評価関数の設定が困難となる

問題点iより，一般的に候補を比較する基準は複数あると考えられる．また問題点iiより，複雑な規則集合ほど，規則集合形成過程において複数の基準に基づく比較が必要であると考えられる．

7 おわりに

本稿では，逆導出原理の問題点に関する考察と，サンプル・プログラムに対する調査に基づき，規則集合獲得では候補を比較することが重要である，との仮説をたてた．これにより，概念とは何か，規則とは何か，という問いに対して，ひとつの示唆を与えた．

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，簡単なプログラム学習に対しては，提案手法の適用可能性を確認した．

今後は，さらなる計算機実験を行ない，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問題点について考察を深めていく予定である．

謝辞

日頃よりご理解を頂き，ご支援下さる，NTTソフトウェア研究所 ソフトウェア基礎技術研究部 後藤滋樹 部長，およびNTT基礎研究所 情報科学研究部 磯崎秀樹 主任研究員に感謝します．

参考文献

[1] Plotkin,G.D.: A Note on Inductive Generalization, *Machine Intelligence 5*, Elsevier North-Holland,New York,pp.153-163(1970).

[2] Plotkin,G.D.: A Further Note on Inductive Generalization, *Machine Intelligence 6*, Elsevier North-Holland,New York,pp.101-124(1971).

[3] Cohen,P.R. and Feigenbaum,E.A.: Learning from Examples, *the Handbook of Artificial Intelligence*, Vol.3,ch.14 D1,Morgan Kaufman(1983).

[4] Buntine,W.: Generalised Subsumption and its application to induction and redundancy, *Artificial Intelligence*, 36(2),pp.149-176(1988).

[5] 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).

[6] Goldberg,D.E.: *Genetic algorithms in Search, Optimization, and Machine Learning*, Addison Wesley Publishing(1989).

[7] 北野宏明: 遺伝的アルゴリズム, 人工知能学会誌, Vol. 7,No. 1,pp.26-37(1992).

[8] 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).