論　文

Technical Papers

帰納学習における帰納論理プログラミングと遺伝的プログラミングの統合

Inductive Learning with Inductive Logic Programming and Genetic Programming

市瀬　龍太郎*　沼尾　正行*
Ryutaro Ichise　Masayuki Numao

* 東京工業大学 大学院 情報理工学研究科 計算工学専攻
Dept. of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology, Tokyo 152-8552, Japan.

1997年10月31日　受理

Keywords: multistrategy learning, inductive logic programming, genetic programming.

Summary

Two approaches to inducing a concept represented in first order logic are inductive logic programming (ILP) and genetic programming (GP). In ILP, concept learning can be considered as a search in the space specified by the background knowledge, and in which the goal concept is represented by Horn clauses. On the other hand, in GP, the search space is specified by terminal and nonterminal symbols, and the goal is represented generally by S-expressions. These two approaches are very similar in terms of their methods and goals, yet their combination in previous work is rare. In this paper, we propose a method that synthesizes the inductive logic programming and genetic programming approaches. The concept behind this approach is to combine the search method of GP, that is, Genetic Algorithm, with the type and mode methods of ILP. We have implemented a system called SYNGIP (SYNthesized system with Genetic programming and Inductive logic Programming) based on the method. Experimental results show that the proposed method can be used to treat, in the same way, learning from training examples that do not have discrete classes, and learning from both positive and negative training examples. Moreover, the proposed method constitutes a novel solution to the closure problem and provides a new bias for concept learning.

1. はじめに

ある状況が与えられた時に，それを元にプログラムを帰納的に作成するシステムとして，大きく分けて2つのアプローチが主にとられてきた．1つはFOIL [Quinlan 90] や Progol [Muggleton 95] に代表される帰納論理プログラミング (ILP：Inductive Logic Programming) [Lavrač 94] のアプローチであり，もう1つは遺伝的プログラミング (GP: Genetic Programming) [Koza 92] のアプローチである．

帰納論理プログラミングは入力として正例，負例，背景知識をシステムに与え，出力としてある概念を示す一階述語論理のサブセットであるホーン節で表現されたプログラムを得る．その際にプログラムの帰納を，背景知識として与えられるリテラルにより構成される空間の探索問題 [Mitchell 82] としてとらえる．探索空間が非常に広大となるため，ゲインヒューリスティック [Quinlan 90] や例や節の記述長 [Muggleton 88] を使った探索手法などが提案され，各々のシステムに特化された探索手法がとられている．一方の遺伝的プログラミングでは入力として適合度，終端記号，非終端記号，パラメータをシステムに与え，出力として学習された概念のプログラムを表すS式を得る．ここであげられる終端記号，非終端記号は帰納論理プログラミングで与えられる背景知識に相当し，適合度は教師信号

として与えられる正例，負例に相当する．遺伝的プログラミングでも帰納論理プログラミングと同様に，終端記号と非終端記号により定義される複雑な概念空間の探索を行うことで目標の概念に相当するプログラムのS式を見つけ出している．その探索の際にはマルチポイントの探索戦略を持つ遺伝的アルゴリズムを用いている．

このように，これらはお互いに共通する機能や目的を持っているにもかかわらず，現在まで2つを統合したシステムに関しての研究はあまり行われてこなかった．これらを統合して扱うことができれば，双方の領域に蓄積されてきた技術，手法などを利用することができ，互いの弱点を克服することができるようになると考えられる．そこで，本稿では一階述語論理レベルの概念学習のタスクに対して，帰納論理プログラミングと遺伝的プログラミングを統合する新しいアプローチを提案し，それに基づくシステムSYNGIP (SYNthesized system with Genetic programming and Inductive logic Programming) について述べる．この手法は，帰納論理プログラミングのタイプとモードに基づく遺伝的プログラミングの枠組であり，遺伝的アルゴリズムに基づいた適応的な探索手法と論理プログラミング的な手法を統合したものである．このことにより，帰納論理プログラミングの学習で用いられる正例，負例の入力に対する学習と，遺伝的プログラミングの学習で用いられる連続値の入力に対する学習を統合化して扱うことができるようになる．さらに，遺伝的プログラミングにおける関数の設計問題に対処し，探索空間を抑制すると同時に，帰納論理プログラミングに新たなバイアスを導入することにもなる．

2章では本稿で提案する統合化のための手法について説明する．**3**章では**2**章で説明をした枠組を用いて，正例，負例で教師信号が与えられるタスクを用いて，本手法の学習能力について述べる．**4**章では連続的な教師信号が与えられるタスクにおいて本手法の学習能力について述べる．**5**章では，他の研究との比較を行い，**6**章で本手法の有効性についてまとめる．

2. 帰納論理プログラミングと遺伝的プログラミングの統合

2・1 ホーン節による探索

遺伝的プログラミングと帰納論理プログラミングの統合化にあたり，両者の定式化の違いについて整理すると，歴史的背景から導出される言語に大きな違いがあることがあげられる．遺伝的プログラミングでは，一般的に導出される言語はLISPのS式であり，帰納論理プログラミングではPrologのホーン節である．導出される言語は機械学習において，非常に大きなバイアスとなることが知られている [Lavrač 94]．S式とホーン節の比較を行うと，ホーン節は探索範囲の低減化について多くの研究 [Muggleton 95, Quinlan 90] がなされていると同時に，単一化により容易に変数同士の関係を扱える点で非常に大きな利点がある．よって，ここではPrologのホーン節を用いて学習を行うことにする．

(1) ランダムに個体の生成を行い初期集団を構成する．
(2) 個体の適合度を計算し，順序づけをする．
(3) 負例をカバーしないで正例をカバーする個体と，適合度がある閾値よりも大きな個体を仮説Hに加える．
(4) 訓練例から仮説Hに加えた個体でカバーされる正例を取り除く．3で仮説Hに個体が加えられた場合には2に戻る．
(5) 適合度が上位の個体に対して，オペレータを施し次の世代の集団を生成する．
(6) 2に戻る．

図1 被覆集合遺伝的アルゴリズム

ホーン節の背景知識により構成される空間を探索する際には，導出される言語と同様に探索手法も大きなバイアスの1つになる．多くの帰納論理プログラミングのアルゴリズムでは貪欲探索を変形させたものを用いて空間の探索を行っているため，簡単に局所解に陥り解が得られなくなってしまう [Lavrač 96]．そこで，本手法では探索を行う手法としてマルチポイントの探索戦略である遺伝的プログラミングの探索手法を用いることにする．

本稿で提案するSYNGIPでは，上記のようなアプローチをとったため，帰納論理プログラミングと遺伝的プログラミングの技術を導入することができる．遺伝的プログラミングでは再帰を含む節の学習において弱点があることが知られている．その原因は基底部と再帰部の両方が一度に進化しなければならない点にある [Wong 96]．一方の帰納論理プログラミングでは，被覆集合アルゴリズム [Quinlan 90] を用いることで，さきに基底部を構成し，それから再帰部を構成するため，再帰を含む節の学習が容易である．そこで遺伝的アルゴリズムに被覆集合アルゴリズムの利点を融合した被覆集合遺伝的アルゴリズム (図1) を提案し，SYNGIPに実装した．

2・2 モードとタイプについて

本稿で提案する手法では，ホーン節の形式に基づく個体を考えているため，帰納論理プログラミングで培われてきた技術である精密化演算子 [Shapiro 82] を利

用することができる．各述語はモードおよびタイプと共に宣言される．それらを利用すると精密化演算子を適用した際に文法的に有効な節のみを生成することができる．この手法は初期集団の生成や突然変異で新たな節が必要な時に使われる．

モードとタイプを利用することで，他にも利点が出てくる．従来の遺伝的プログラミングではどんな変数やデータも使用できるように，閉包性が必要とされてきた [Koza 92]．しかし，本手法では，モードとタイプを利用することで，変数やデータの扱い方が規定されるため，すべての型を受理できるように述語を設計しなくてもよい．このことにより，学習される概念を記述する述語の設計が容易になると同時に，遺伝的プログラミングで生成される無意味な個体群を自動的に排除し，探索の範囲を狭められる．

述語の設計が容易になることを示そう．たとえば，ある部品リストからあるパーツに隣接するパーツのリストを取り出す述語を考えると，本手法では背景知識に $next\_parts(+parts, +[parts\_list], -[parts\_list])$ のようにモードとタイプの宣言を書くだけで，変数の入出力とタイプが規定されるため，異なる属性の変数同士をうまく分離することができる．従来の遺伝的プログラミングの手法では，第1引数に部品リストがきた場合，第2引数に部品がきた場合などを配慮しながら述語の設計を行わなければならない．これに数字などの他の属性が入ると引数の種類 m，引数の数 n として m^n の入力に対し配慮をしながら関数の設計を行わなければならないが，本手法を用いるとこのような問題は起こらない．

(1) 親1，親2に対しリテラル同士の区切りを交叉点として交叉点をそれぞれランダムに決定する．
(2) 親1のヘッド部分および交叉点前のリテラルと，親2の交叉点より後ろのリテラルを接合する．
(3) 親1の交叉点の前までの変数の集合 $V1$ を計算する．
(4) 親2の交叉点より後ろの入力変数の集合 Vi，出力変数の集合 Vo，Vo から Vi の要素を取り除いた Vd を計算する．
(5) 親2のヘッドにある入力変数が，集合 Vi の中にある場合には，その変数を親1のヘッド上の同じ位置の変数と単一化させ，Vi から取り除く．
(6) Vi にある変数すべてに対して，型が同じで $V1$ 中に存在する変数と単一化させる．
(7) Vd にある変数をランダムで選択し，型が同じで $V1$ に存在する変数と単一化させ，得られる節を子供の個体とする．

図2 リテラル同士の交叉アルゴリズム

```
head(+[x],+[x],-[x])
literal1(+[x],-[x])
literal2(+[x],-[x])
literal3(+[x],+[x],-[x])
literal4(+[x],-[x])
literal5(+[x],+[x],-[x])
```

図3 与えられるモードとタイプ

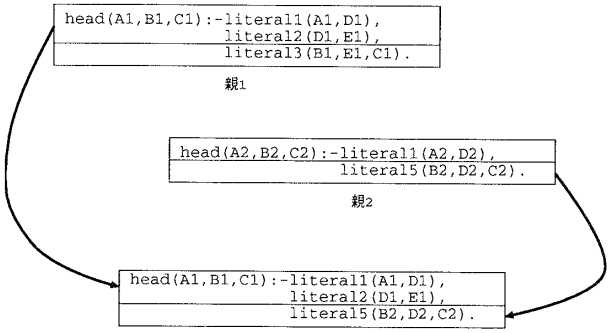


図4 単純な交叉を用いた場合

2・3 遺伝オペレータ

ホーン節に遺伝的アルゴリズムを適用するために個体を定義しなければならない．本稿では，個体の定義は仮説として意味のある節の1つ以上の集合ととらえる．その際に，リテラルを遺伝子としてとらえ，リテラル単位で遺伝オペレータが施されるものと，リテラルを一まとめにした節を遺伝子としてとらえ節単位で遺伝オペレータが施される場合の2つを考える．遺伝オペレータとしては交叉と突然変異のみを考える．

〔1〕 個体間の交叉

リテラル同士に対して交叉を行う際には，新しい個体の生成を行う場合に，切断した個体同士を単純に接合するだけでは前半部のスキーマと後半部のスキーマは保存されるが，前後の連続性がなくなってしまう．そこで，帰納論理プログラミングで用いられるモードとタイプの制約を用いて引数部分の修正を行う．そのアルゴリズムを図2に示す．アルゴリズムの途中で得られる節がヘッド部分だけになったり，単一化する候補がなくなったりした場合には最初からやり直す．また，単一化の候補が複数あった場合には，ランダムで選択される．

このアルゴリズムを例を用いて説明する．図3が与えられたモードとタイプであり，head (+[x],+[x],−[x]) というのは述語 head は第1引数と第2引数が x という属性を持つリストの入力であり，第3引数は x という属性を持つリストの出力であることを示す．このような変数をそれぞれ入力変数，出力変数と呼ぶ．従来の遺伝的アルゴリズムで用いられていたような単純な交叉手法を用いると，図4のように親1と親2の中でランダムに交叉点を決め接合することとなる．この図中では，親1の literal2 と literal3 の間と，親2の

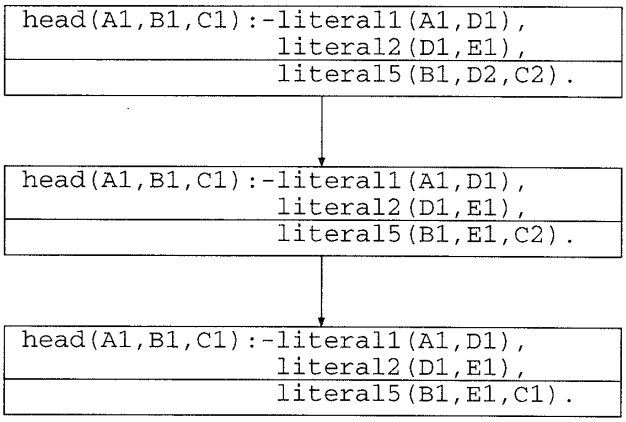


図5 引数の修正

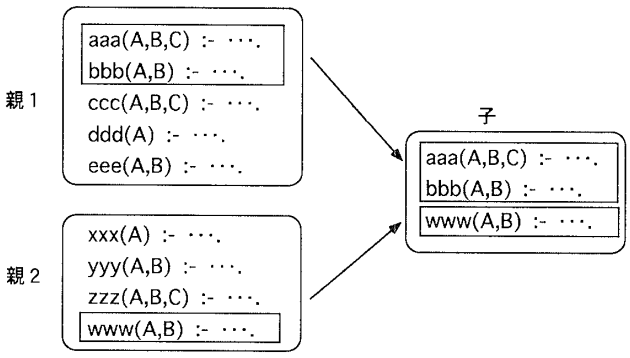


図6 節同士の交叉の例

literal4 と literal5 の間が交叉点となっている．しかし，この結果得られた個体は引数が異なるため，前後の連続性が保たれなくなる．そこで，図5のように引数の単一化を順次行うことで解決をする．モードを利用し，親1から遺伝する部分の変数の集合と親2から遺伝する部分のアルゴリズム中の変数の集合をそれぞれ計算する．ここでは親1から遺伝する部分の変数の集合が A1,B1,C1,D1,E1 となり，親2から遺伝する部分の入力変数の集合が B2,D2 である．出力変数の集合が C2 となる．ここで，B2 に着目すると，B2 は親2のヘッドに出現するので，親1の同じ位置の B2 が B1 に単一化することになる．この変数を取り除くと，集合の中に D2 だけが残る．D2 はランダムでどれかと単一化することになる．ここでは E1 が選択されたとすると，D2 は E1 と単一化する．それが図5の中段の節である．次に，C2 に対しても同じような処理が行われ，ランダムで C1 が選択された結果，C2 が C1 と単一化し，最終的に子となる節が得られる．それが，図5の下の節である．C2 に対しては，ランダムにより単一化されない場合もある．

節同士で交叉をする場合には，普通の遺伝的アルゴリズムと同様に子供となる個体を生成することができる．この交叉の様子を例示したのが図6である．図6の例では親1の2番目の節の後と，親2の4番目の節の前が交叉点となり，親1の前半部と親2の後半部が接合することにより新しい子供となる個体を生成している．

〔2〕 個体の突然変異

交叉の場合と同様に，突然変異を起こす場合にリテラルに対して行う場合とリテラルのまとまりである節に対して行う場合が考えられる．リテラルに対して行う場合には，変化させた後で，引数の数や引数の属性が必ずしも前のリテラルと同じになるとは限らない．従って，引数に整合性が保たれるようにしなければならない．交叉で用いたアルゴリズム (図2) と同様にリテラルの入出力に着目することにより，この問題に対処する．リテラルのまとまりである節に対して突然変異を起こす場合には，前述の精密化演算子を用いることで構文上，入出力や属性が正しい節のみを生成し，既存の節と交換することで突然変異を行う．

2・4 教師信号

遺伝的プログラミングの手法では教師信号として連続値が用いられ，帰納論理プログラミングの手法では正例と負例という離散値が用いられる．従来の帰納論理プログラミングの学習手法では，連続的な評価値の教師信号を持つドメインにおいては，正例と負例をある閾値をもとに分類して用いていた [Numao 97]．その際には，閾値を予め人間が決めなければならないため，明らかに閾値が分かるような場合以外は帰納論理プログラミングで学習を行うのは困難であった．また，FOR[Karalič 97] のように変数として連続値を利用する方法は提案されているが，教師信号として連続値を利用するようなドメインの学習には利用できない．そこで，連続値による教師信号について学習をすることを考え，離散値の教師信号は連続値に変換することで利用する．

〔1〕 適合度関数

正例と負例だけの離散値の教師信号は入力された場合に，正例と負例から連続値の教師信号となる適合度を計算する関数として，次の関数を導入する．

$$Fitness(I) = \left(\frac{p}{N_p} \times \frac{N_n - n}{N_n}\right) + \alpha(L_{max} - L) \quad (1)$$

式 (1) に出現する文字は，p が個体 I の被覆する正例の数，n が個体 I の被覆する負例の数，N_p が全正例の数，N_n が全負例の数，L_{max} が許されるリテラル数

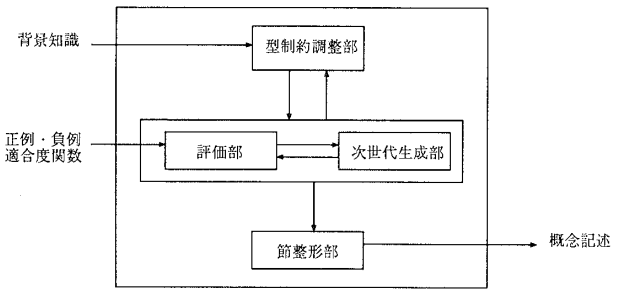


図7 SYNGIPの構成図

の最大値，Lが個体Iのリテラルの数，αが節の長さの考慮の度合いを表すパラメータをそれぞれ表している．式(1)では，正例が満たされ，負例を満たさないような個体に対して評価が高くなるようになっている．これにより正例と負例のみを与えられた場合に適合度を計算することができる．

連続値の教師信号を与えられた場合には，その教師信号自体を適合度関数にすることができる．この場合は，その教師信号自体を適合度関数とする．

2・5 システムの構成

以上で述べた学習手法を実際に計算機上に実現したシステムSYNGIPにより，学習実験を行った．SYNGIPの入力として，背景知識と教師信号となる正例，負例または適合度関数を与え，SYNGIPの出力として，学習された概念記述を得る．SYNGIPの構成を図7に示す．各部の役割は次のようになっている．

型制約調整部 背景知識の型の制約に基づき，構成できる節を生成したり，制約に合わない部分の誤りを修正する．

評価部 個体をその適合度関数に従って評価し，適合度を計算する．

次世代生成部 交叉，突然変異を行うことにより次世代の個体を生成する．必要に応じて，型制約調整部を呼び出し新しい節の生成および修正を行う．評価部と次世代生成部が繰り返され，ループが停止するとその時に保持する仮説が節整形部に送られる．

節整形部 ポストプロセッシングとして，余分な節を取り除くなどの節の簡略化を行い，概念記述を出力する．その際には，一般的にILPで用いられる手法と同様の手法 [Lavrač 94] を用いる．具体的には，学習が行われた後の仮説Hに含まれる節Cを1つ取り除いた仮説H'を作り，仮説Hと仮説H'を訓練例を用いて比較した際に，仮説H'が訓練例を説明するための記述として仮説Hと変わらない時，仮説H'を新たな仮説Hとして採用する．そして，仮説H内のすべての節Cに対して，この作業を行い，得られた仮説Hを学習結果として出力する．

システムはSICStus Prolog Version 3[*1]で実現され，Ultra Sparc 1 / model 170 互換機 (SunOS 5.5.1) 上で実験を行った．

3. 正例と負例による学習

まずはじめに帰納論理プログラミングでよく研究されている設定として，正例，負例と背景知識をシステムに入力して概念の学習を行わせ，SYNGIPの能力を調べた．ここではオペレータの適用をリテラル単位とし，帰納論理プログラミングのテストでしばしば用いられ，再帰の概念を含むmember(正例22個，負例19個)とcan_reach(正例19個，負例62個)の概念 [Quinlan 90] の獲得について，実験を行った．個体数は100とし，個体の選択にはトーナメントサイズ2のトーナメント方式を用いた．初期集団を作成する際には**2章**で触れた精密化演算子を用いてできるものの中からランダムに個体の選択を行った．次世代の個体を生成する際の手法にはエリート戦略を用い，世代間ギャップは0.9とした．また，L_{max}の値は15とした．この実験では突然変異は用いなかった．遺伝的アルゴリズムは確率的な探索手法であるため，各々10回ずつ試行した．

その結果，memberとcan_reachの概念の学習は両方とも10回中，10回正確な概念の学習を行うことができた．memberの学習にかかった平均時間は23.20秒，can_reachの学習にかかった平均時間は4.15秒だった．かかった世代数は，どの実験においても高々数世代程度であった．

この実験では突然変異は用いなかった．しかし，変数は交叉の際にランダムに修正されるため，初期集団の中のリテラル内ですべての変数名が異なるリテラルが1つでも存在すれば，どのような変数割当てのリテラルも交叉のみで生成することができる．また，初期集団にこのようなリテラルが全く含まれない確率は極めて小さい．よって，突然変異を用いなくても，ここでは問題にはならないと言える．

この実験により，遺伝的プログラミングでは困難であった再帰的述語を用いた概念に関しても，本稿で説明した手法では学習できることが示された．また，遺伝的アルゴリズムは確率的な探索手法であるため，目

*1 ©Swedish Institute of Computer Science

的の解に毎回必ず到達するという保証はない．しかし，この実験ではすべての場合において正解に到達できている．これはどの初期集団にもすべての述語があり，リテラル内の変数がすべて異なるリテラルが1つ以上あったという条件を満たしていたためと考えられる．

4. 連続値の教師信号による学習

次に連続的な評価値を持つドメインにおいて，本稿で説明した手法の有効性を調べる．この実験ではオペレータの適用の単位をリテラルのまとまりである節とした．また，このドメインにおける学習は，導出する節に再帰が含まれないため，被覆集合型のアルゴリズムを用いる必要性はない．よって，従来の遺伝的アルゴリズムと同様のアルゴリズムを用いた．連続的な評価値を得ることができるドメインとして，電気回路の部品の配置を行う規則の学習を例にとって実験を行った．

部品点数29個，バスの数14個の回路図に対し，ネット情報だけを取り出し，その接続関係を例として与えた．そして，部品を一直線上に並べ，部品の配線長が最短となるような規則の学習を行う．そのため，適合度関数は配線長とし，節を構成する際に使われる「部品Xには端子が1つしか接続されていない」などを示す14種類の背景知識をシステムに与えた．また，節を構成する背景知識とは別に，回路の配線長を計算する知識と部品を配置する順番をシステムに与えた．部品を配置する順番は接続本数の多い部品の順とした．実験に用いた個体数は100とした．初期集団の生成と選択の手法は**3**章の実験と同様な手法をとった．その他のパラメータは表1の通りとした．学習により獲得された概念に対してどの程度の学習効果があるかを調べるため，学習された概念を別の回路図に適用し，配線長を調べた．概念を学習する前に与えられた知識*2を用いると，配線長は訓練例として与えた回路では991，テスト例として与えた回路では2775になる．

図8は世代を重ねた時に各々の個体で配線した時の配線長の推移を表したものである．どの条件においても，世代数を重ねることにより訓練例の配線長が短くなる規則を学習していて，評価が連続値を持つドメインに関しても学習が行われていることが分かる．このようなドメインでは，前述したように従来の帰納論理プログラミングを適用することは困難であった．

図9は200世代目で学習された知識を用いた時の配線長，および得られた知識を訓練例とは別の回路で配線に使用した時の配線長である．図9を見るとどの条件においても，汎用性のある正しい概念が学習できていることが分かり，本手法が有効であったと言える．付録に本手法で学習された節の例を載せておく．

この問題を従来の部品配置問題としてとらえた時に，GAを使った配置手法として吉川らの手法 [吉川 95] があげられる．しかし，本手法は配置の最適化を行っているのではなく，配置の規則自体を学習している点で，従来にはないアプローチであると言える．

この実験では，もし遺伝的オペレータを交叉のみに限定すると，初期集団の中に解として使われる節がすべて入っていないと目的のプログラムは生成することができない．しかし，ここでは突然変異があるため，初期集団でどのような状態にあっても，どのようなProlog

表1 実験の条件

方法	トーナメントサイズ	世代間ギャップ	突然変異率
方法1	2	0.9	0.05
方法2	5	0.9	0.05
方法3	2	0.7	0.05
方法4	2	0.9	0.10

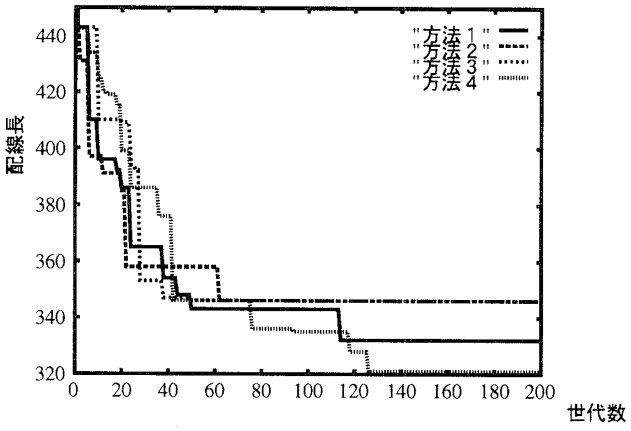


図8 配線長の推移

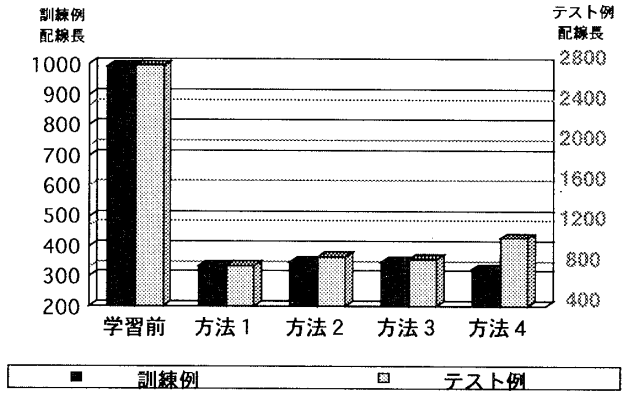


図9 配線結果

*2 入力された順番通りに並べるという知識．

プログラムも生成できる．

5. 関 連 研 究

Varšek の研究した GILP [Varšek 95] では，ホーン節を引数の次数や変数の数の最大値などに基づいて一度文字列に変換したり，ホーン節を木構造に変換したりする．そして，その変換されたものに対して遺伝的アルゴリズムと同じ手法を用いることで，目標となる概念に相当するホーン節の探索を実現している．そのため，変換する過程で非常に多くの無駄なものを生成することになる．本研究で提案した手法では，Prolog で用いられるホーン節をそのまま個体として見なしている点が，GILP とは全く異なり，飛躍的に効率が改善されている．また，GILP は遺伝的アルゴリズムを導入しているにもかかわらず，学習できる問題のクラスとしては帰納論理プログラミングで用いられる正例と負例で表現されるクラスにしか対応できない．本研究では遺伝的アルゴリズムを用いることで，遺伝的プログラミングの学習手法を統合し，学習できる問題のクラスの拡張を行っている．

Wong らの研究 [Wong 95a, Wong 95b, Wong 96] はホーン節や S 式などを表現することができる論理文法を定義し，その論理文法から導出された木構造の言語で概念が表現される．その論理文法は文脈自由文法を一般化した確定節文法と同じ記述法が用いられている．この文法から導出される木に対し，遺伝的プログラミングと同じ手法を用いることでホーン節などの学習を行うことに成功している．しかし，変数間の依存関係が必要であるホーン節の学習の際には，この文法上で変数間の関係に相当する文脈の関係が適切である必要性があり，もし不適切であると文脈自由文法に基づくため，学習されるべき概念の変数関係が表現できなくなってしまう．つまり，変数の関係について予測しながら，概念を表現する文法の設計をしなければならないという弱点がある．本手法ではモードとタイプの宣言を用いることで，自動的に変数間の関係を構成することができる．また，本手法は帰納論理プログラミングの型の制約に注目し，ホーン節自体に遺伝的アルゴリズムの手法を取り込んでいるのに対し，Wong らの構築した LOGENPRO [Wong 95a] をはじめとするシステムはホーン節をある文法を用いることで，木構造として表現し，プログラムの合成を行おうとしている点で全く異なるアプローチであると言える．

Montana の研究した STGP [Montana 94] は遺伝的プログラミングに対して型の制約を採用し，交叉の際にその制約を利用している．本研究では帰納論理プログラミングで用いられるタイプとモードを用いているため，精密化演算子と同様の手法を利用することで，突然変異の際にも柔軟に節中の変数の修正を行うことが可能となっている点で異なっている．また，本手法を用いると正例，負例からの学習もできるという点で学習できる問題のクラスが STGP よりも拡張されている．

6. お わ り に

本研究では，帰納論理プログラミングの長所と遺伝的プログラミングの長所はお互いに排他的であるわけではないということに着目し，その 2 つの手法を融合した頑健な学習システムを構築した．節の構成手法として帰納的論理プログラミングの枠組みを，探索のための手法として遺伝的アルゴリズムを用いることを提案し，それに伴いホーン節における交叉，突然変異などの手法の定義を行った．この手法に従ってシステム SYNGIP を実装し，次のことが分かった．

本稿で説明したシステムは簡単なドメインに対しては，従来の帰納論理プログラミングと同様の動作を行い，遺伝的アルゴリズムを用いた本手法が貪欲法を用いた従来の帰納的論理プログラミングシステムの代わりに用いるのに十分な能力があることを示せた．また，再帰の入っている概念に関しても学習ができた．さらに評価が連続的に得られるような，従来の帰納論理プログラミングの手法では学習を行うことが困難であったドメインに対しても，学習を行うことができることを実験で示した．帰納論理プログラミングの型の制約を利用することで，探索空間を小さくすると同時に，遺伝的プログラミングにおける閉包性の問題を解決することにも成功した．

謝　　辞

本研究の一部は情報処理振興事業協会 (IPA) が実施している「創造的ソフトウェア育成事業」の援助によるものである．

◇ 参 考 文 献 ◇

[Goldberg 89] Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison Wesley (1989).

[Karalič 97] Karalič, A. and Bratko, I.: First Order Regression, *Machine Learning*, vol.26, pp.147-176, Kluwer Academic Publishers (1997).

[Koza 92] Koza, J. R.: *Genetic Programming*, MIT Press

(1992).
[Lavrač 94] Lavrač, N. and Džeroski, S.: *Inductive Logic Programming Techniques and Applications*, ELLIS HORWOOD (1994).
[Lavrač 96] Lavrač, N., *et al.*: Handling Imperfect Data in Inductive Logic Programming, in *Advances in Inductive Logic Programming*, pp.48-64, IOS Press (1996).
[Mitchell 82] Mitchell, T. M.: Generalization as Search, *Artificial Intelligence*, vol.18, pp.203-226, North-Holland (1982).
[Montana 94] Montana, D. J.: Strongly Typed Genetic Programming, *BBN Technical Report*, No.7866 (1994).
[Muggleton 88] Muggleton, S. and Buntine, W.: Machine Invention of First-order Predicates by Inverting Resolution, in *Proc. of 5th Machine Learning Conference*, pp.339-352, Morgan Kaufmann (1988).
[Muggleton 95] Muggleton, S.: Inverse entailment and Progol, *New Generation Computing*, vol.13, pp.245-286, Ohmsha (1995).
[Numao 97] Numao, M., *et al.*: Acquisition of human feelings in music arrangement, in *Proc. of the 15th International Joint Conference of Artificial Intelligence*, pp.268-273 (1997).
[Quinlan 83] Quinlan, J. R.: Learning efficient classification procedures and their application to chess end games, in *Machine Learning: an artificial intelligence approach*, Tioga (1983).
[Quinlan 90] Quinlan, J. R.: Learning Logical Definitions from Relations, *Machine Learning*, vol.5, pp.239-266 (1990).
[Quinlan 92] Quinlan, J. R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann (1992).
[Shapiro 82] Shapiro, E. Y.: *Algorithmic Program Debugging*, MIT Press (1982).
[Varšek 95] Varšek, A.: Genetic Inductive Logic Programming, *Technical Report*, ESPRIT III project (1995).
[Wong 95a] Wong, M. L. and Leung, K. S.: An Induction System that Learns Programs in different Programming Languages using Genetic Programming and Logic Grammars, in *Proc. of the 7th IEEE International Conference on Tools with Artificial Intelligence*, pp.380-387 (1995).
[Wong 95b] Wong, M. L. and Leung, K. S.: Inducing Logic Programs with Genetic Algorithms : The Genetic Logic Programming System, *IEEE Experts*, vol.9, pp.68-76 (1995).
[Wong 96] Wong, M. L. and Leung, K. S.: Evolving Recursive Functions for the Even-Parity Problem using Genetic Programming, in *Advances in Genetic Programming*, vol.2, pp.221-240, MIT press (1996).
[吉川 95] 吉川大弘, 古橋武, 内川嘉樹：プリント基板における部品自動配置問題に適したコーディング手法, 電学論 D, vol. 115, No. 5, pp.642-651 (1995)

〔担当委員：小林重信〕

◇ 付　　録 ◇

A. 4章で学習した知識の例

4章で200世代目に得られた節の1つを例として以下にあげる.

$$place(X,Y,Z) :- select\_parts1(X,Y,W),$$
$$insert\_between(X,W,Y,Z),$$
$$many\_legs(X).$$

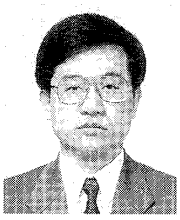
*place*は Y の配列状態に対して X の配置を行い, Z の状態にするという述語. *select_parts1*は2段で X に接続されているパーツで他に接続している端子の数が最大のパーツ W を選択する述語. *insert_between*は Y の配置状態に対して, X をパーツリスト W の間に配置を行い Z の状態にするという述語. *many_legs*は X が他に接続している端子をたくさん持つことを示す述語.

著　者　紹　介

市瀬　龍太郎(学生会員)

1997年3月東京工業大学大学院情報理工学研究科計算工学専攻修士課程修了. 現在, 同大学院博士課程在学中. 機械学習の研究に従事. 電子情報通信学会, 日本認知科学会各会員.

〈ichise@cs.titech.ac.jp〉



沼尾　正行(正会員)

1982年東京工業大学工学部電気電子工学科卒業. 1987年同大学大学院情報工学専攻博士課程修了. 工学博士. 東京工業大学工学部情報工学科助手, 講師を経て同大学大学院情報理工学研究科計算工学専攻助教授. 1989～1990年Stanford大学客員研究員. 人工知能, グローバル知能, 機械学習などの研究に従事. 情報処理学会, 日本認知科学会, 日本ソフトウェア科学会, AAAI, AIUEO各会員.

〈numao@cs.titech.ac.jp〉