並列論理型言語 KL1 を用いた並列帰納学習システムの作成*

1P－5

川野辺 甲　大和田 勇人　溝口 文雄†

東京理科大学　理工学部‡

1　はじめに

近年，機械学習の分野で帰納的推論を取り入れ，背景知識と事例から一般化規則を導出する際に，用いられた概念を論理プログラムで生成する研究が盛んに行なわれている．Muggleton の GOLEM[1] や Quinlan の FOIL[2] などが代表的である．一般に，これらの帰納学習システムは正事例集合 E^+，負事例集合 E^-，背景知識 K を用いて，$K \cap H \vdash E^+$ かつ $K \cap H \nvdash E^-$ を満たす規則 H を導出する．しかし，この様な規則を効率良く導出するための方法は，まだ確立されていない．そこで，本稿は数ある帰納学習システムの中でも，h-easy の概念を導入して，有限の長さで他のアルゴリズムで学習できない多くの論理プログラムを帰納学習することのできる GOLEM を取り上げ，GOLEM の基本的アルゴリズム (greedy アルゴリズム) の問題点に触れる．さらに，その問題を解決する手段として並列処理の考えを導入した並列帰納学習システム（PAG）について述べる．これは Plotkin が提案した相対最小一般化（RLGG）[3] と関数の依存関係を用いたリテラルの削減といった基本的な手法に，並列処理を導入することで，有効解が放棄されることなく得られるシステムである．

2　greedy アルゴリズムの問題点

このアルゴリズムは，各正事例のペアに対して RLGG を行ない，得られた仮説（C_i）に対し，" 一番正事例を満たすものを C とする" という制約を用いて，他の有効な仮説をその時点で放棄してしまう可能性に問題がある．

本システム（PAG）では，できるだけ有効な仮説を保持するために，得られた仮説の中からただ一つ選択するのではなく，正事例を含む数が多いものから順に k 個選択している．そして，これらの仮説を全て考慮するために並列処理を導入した．並列処理を導入し，有効な仮説をできるだけ放棄せずに得ることによって，より良い規則を得ることができた．

図 1 は greedy アルゴリズムが仮説 C_i の中からただ一つ C を選択し，PAG のアルゴリズムでは C_i の中から k 個の仮説を選択する様子を図示している．なお，*は正事例を示している．

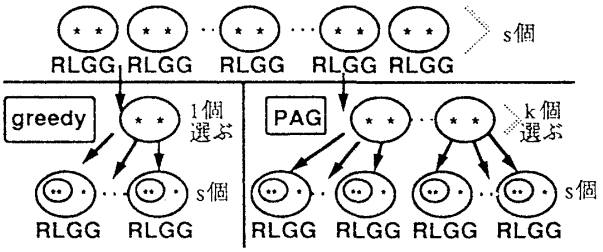

図 1 : greedy アルゴリズムと PAG の仮説の選び方

3　並列帰納学習システム：PAG

本システムは，得られた仮説候補をそれぞれ独立に処理し，最後に一つにまとめ一般化規則を求めるので，高い並列効果が期待できる．また，並列処理を行なうための言語に，ICOT で開発された KL1 を用いた．これは，単純であるにも関わらず並列プログラムに関して，かなりの表現力を有している優れた言語である．

3.1　動的負荷分散制御

PAG のアルゴリズムは，分散する負荷の数が動的に変化するため，静的負荷分散制御では効率の良い負荷分散を行なうことができない．そこで，並列処理を行なうための負荷分散制御として，動的負荷分散制御を用いた．以下にその概要を示す．

1. プロセッサ 0 に暇なプロセッサを管理するマネージャを置き，負荷分散を行なう処理も行なわせる．
2. 各プロセッサに割り付けられた部分問題の処理が終了すると，プロセッサ 0 に結果を返し，次に処理する部分問題がなければ暇であることをプロセッサ 0 に知らせる．
3. プロセッサ 0 では，部分問題の処理結果をもとに新たな部分問題を，暇プロセッサ管理マネージャの情報に

*Parallel inductive learning system using KL1
†Masaru Kawanobe,Hayato Ohwada,Fumio Mizoguchi
‡Faculty of Sci. and Tech., Science Univ. of Tokyo

基づいて各プロセッサに割り付ける.

3.2 並列機能学習システムのアルゴリズム

並列機能学習システムのアルゴリズムを以下に示す.S2 から S4 は並列に行なわれる.

[S0] 正事例の集合をε^+, 負事例の集合をε^-, 背景知識を Kとする.

[S1] $\exists e_0 \in \varepsilon^+, \exists e_i \in \varepsilon^+,\ C_0 = e_0 \leftarrow K, C_{i,1} = e_i \leftarrow K$ に対して $C_i = rlgg(C_0, C_{i,1})$ なる C_i を求める $(i = 1, \cdots, s)$.

[S2] C_i に対して節の削減をし, 正事例を多く満たす順に k 個選ぶ. $(C_j, j = 1, \cdots, k)$, C_j を作った正事例のペアを $\{e_0, e_j\}$ とする.

[S3] $\varepsilon^{+\prime} = \varepsilon^+ - \{e_0, e_j\}$,$\exists e_l \in \varepsilon^{+\prime},\ C_{l,1} = e_l \leftarrow K$ に対して $C_l = rlgg(C_j, C_{l,1})$ なる C_l を求める $(l = 1, \cdots, s)$.

[S4] C_lに対して節の削減をし, 正事例を多く満たす順に k 個選ぶ $(C_m, m = 1, \cdots, k)$.

1) もし, $C_m = \phi$ なら C_j を仮説として終了する.

2) もし, 正事例を満たす数が C_j より C_m の方が少なければ C_j を仮説として終了する. そうでなければ, S2 から繰り返す.

[S5] 得られた仮説集合の中から, 最も正事例を含むものを Cとする. $C \vdash \varepsilon_s^+, \varepsilon_s^+ \subset \varepsilon^+$の時, もし, $\varepsilon - \varepsilon_s^+ = \phi$ なら, C を求める一般化規則として終了する. そうでなければ, C を一般化規則として保持し, $\varepsilon^{+\prime\prime} = \varepsilon - \varepsilon_s^+$ となる $\varepsilon^{+\prime\prime}$ を新たな正事例の集合として, S0 から繰り返し, 別の一般化規則を求める.

全解探索では, 例題の規模によって並列処理を用いても, 探索空間が膨大になり対処不可能となる. そこで, PAG では例題の規模が大きい時, RLGG を求めるペアの作成方法に制約を付けたり, 例の数によって, S2 の C_j の個数を変えて, 探索空間をある程度の規模に保っている.

4 性能比較

表 1 に member と qsort について GOLEM と PAG の結果を示す. GOLEM は sun/4, PAG は Multi-PSI(16PE 版) 上で実行させた結果である. また, BG は背景知識である. さらに GOLEM と PAG で得られた member と qsort の一般化規則を示す.

表 1: 例題の結果

Problem	Numbers of Exp.	Numbers of BG	GOLEM (sec.)	PAG (sec.)
member	14	20	2.4	2.3
qsort	15	79	22.7	84.6

```
(GOLEM)
member(A,[B,C|D]) :- member([A,[C|D]).
qsort([A|B],[C|D]) :-
        qsort(B,E),part(A,E,F,G),
        app(F,[A|G],[C|D]).

(PAG)
member(A,[B,C|D]) :- member(A,[C|D]).
qsort([A|B],[C|D]) :-
        part(A,B,E,F),qsort(E,E),
        qsort(F,G),app(E,[A|G],[C|D]).
```

以上の結果を見ると, PAG の一般化規則を得るスピードが member は GOLEM と同程度であったが,qsort は GOLEM の方が速い. しかし, PAG で得られた一般化規則は, GOLEM より真の qsort のプログラムに近いものであった.

5 まとめ

本稿では, greedy アルゴリズムが有効な仮説を捨てるという問題点に対して並列処理を取り入れることで, 有効な仮説をできるだけ捨てることなく, 有効な一般化規則を効率良く求めることができた. 今後の課題として, システムのスピードアップや, 例題が増加した時の探索空間を制限する方法などを検討する必要がある.

参考文献

[1] S.Muggleton,C.Feng:Efficient induction of logic programs.

In New Generation Computing,1991,Vol8,pages 42-61,1990

[2] I.R.Quinlan:Determinate Literals in Inductive Logic Programming. *Proc.of.the Eighth International Machine Learning Workshop*,PP.442-446(1991).

[3] G.D.Plotkin:A note on inductive generalization. *In B.Meltzer and D.Michie, editors,Machine Intelligence 5*,PP.153-163 Elsevier North-Holland,New York(1970)