電子情報通信学会論文誌 2002/6 Vol. J85–D–I No. 6

ILP アルゴリズム FOIL を並列化する三つの手法とその比較

松井藤五郎† 犬塚 信博††（正員）
世木 博久†††（正員）

A Comparison of Three Methods for Parallelizing an ILP Algorithm FOIL

Tohgoroh MATSUI†, *Nonmember*, Nobuhiro INUZUKA††, and Hirohisa SEKI†††, *Regular Members*

† 名古屋工業大学大学院工学研究科，名古屋市
Graduate School of Engineering, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya-shi, 466–8555 Japan
†† 名古屋工業大学共同研究センター，名古屋市
Center for Cooperative Research, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya-shi, 466–8555 Japan
††† 名古屋工業大学知能情報システム学科，名古屋市
Department of Intelligence and Computer Science, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya-shi, 466–8555 Japan

あらまし 本論文では，ILP アルゴリズム FOIL を並列化する手法について比較する．ILP を並列化するには，探索空間分割，背景知識分割，事例集合分割の三つの方法がある．これらについて，並列計算機上での実験結果を示すとともに，プロセッサの待ち時間，繰返しごとの処理，FOIL 以外の ILP アルゴリズムへの応用，という三つの視点から比較した．その結果，ILP を並列化するには一般に事例集合を分割するのが望ましく，FOIL の並列化に限ると背景知識を分割するのが望ましいとの結論を得た．

キーワード 帰納論理プログラミング，並列処理

1. まえがき

帰納論理プログラミング（inductive logic programming: ILP）は論理プログラミングの枠組みで機械学習を行う．ILP は一階述語論理に基づいた強力な表現能力をもつが，そのため学習速度が遅くなってしまう．そこで，近年，ILP アルゴリズムを並列化する研究が行われている [1], [2]．本論文では，ILP アルゴリズム FOIL [3] を分散メモリ型並列計算機上で並列化する三つの手法と実験結果を示す．最後に，異なる三つの視点からこれらを比較し，どの手法が適するかを述べる．

FOIL は，属性–値学習の研究で開発された決定木学習アルゴリズム C4.5 をもとに，1 階述語論理を扱えるように拡張した ILP アルゴリズムである．表 1 に FOIL アルゴリズムの概要を示す．FOIL に与えられる事例は $R(c_1, \cdots, c_n)$ の形をしており（R は関係名，c_i は定数），このアトムの組 $\langle c_1, \cdots, c_n \rangle$ をタプルと呼ぶ．背景知識も一般の節形式ではなくそれぞれの関係に属するタプルの集合として与えられる．FOIL の探索は，右辺がない（最も一般的な）ホーン節からはじめ，その節が負事例をカバーしなくなるまで右辺にリテラルを追加して節を特殊化する．ある特殊化中の節が変数 $v_1, \cdots, v_x$ を含んでいるとき，事例集合はこれらの変数への代入例 $\langle c_1, \cdots, c_x \rangle$ の集合として表現される．特殊化した節の評価は，追加されたリテラル L が $R(v_{i_1}, \cdots, v_{i_r})$ のとき，特殊化前の事例集合 T と関係 R に属するタプルの集合 B^R を結合して新しい事例集合 $T' = T \bowtie B^R$（$\langle c_1, \cdots, c_{x'} \rangle \in T'$ は $\langle c_1, \cdots, c_x \rangle \in T$ かつ $\langle c_{i_1}, \cdots, c_{i_r} \rangle \in B^R$ を満たす）を求め，T と T' の情報量の差に基づいた関数 $gain$ を用いて行う．

$$gain(L) = s \times \left(\log_2(T'^+/|T'|) - \log_2(T^+/|T|) \right)$$

ここで，T^+ は T に含まれる正事例の数，s は $T' = T \bowtie B^R$ に展開された正事例の数を表す．目標関係 R_t の負事例をカバーしない節が探索されると，その章がカバーする正事例を事例集合から除く．これらをすべての正事例がカバーされるまで繰り返す．

表 1 FOIL アルゴリズムの概要
Table 1 An outline of FOIL algorithm.

$\mathcal{P}$:= 空プログラム
$\mathcal{T}$:= 目標関係 R_t の正事例の集合
While $\mathcal{T} \neq \phi$
  C := “$R_t(v_1, v_2, \cdots) \leftarrow$”
  While C が R_t の負事例をカバーする
    適当なリテラル L を見つけ，C の右辺に加える
  C によってカバーされた事例を $\mathcal{T}$ から取り除く
  C を $\mathcal{P}$ に加える

2. FOIL を並列化する三つの手法

FOIL の計算の大部分は特殊化した節を評価するための結合演算である．これを複数のプロセッサに分散して実行するには，次の三つの方法がある．以下，分割された処理を行うプロセッサの台数を n とする．

探索空間分割：(1) 評価の対象となる節の集合 $\mathcal{C} = \{C_1, \cdots, C_k\}$ を互いに素かつできる限りサイズの等しい n 個の集合 $\mathcal{C}_1, \cdots, \mathcal{C}_n$ ($\mathcal{C} = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_n$) に分割する．(2) $\mathcal{C}_i$ に含まれる節をプロセッサ i で評価し，$\mathcal{C}_i$ の中の最良の節 $C_{best,i}$ を選択する．(3) 各プロセッサの $C_{best,i}$ を一つのプロセッサに集め，$\mathcal{C}$ の中の最良の節 C_{best} を選択する．

背景知識分割：あらかじめ B^R を互いに素かつできる限りサイズの等しい n 個の集合 $B^R_1, \cdots, B^R_n$

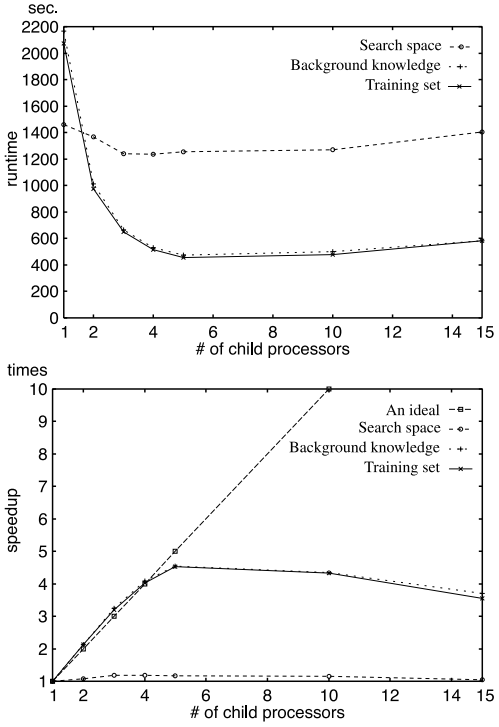

図 1　実行時間と速度向上比
Fig. 1　The run-time and the speed-up ratio.

($B^R = B_1^R \cup \cdots \cup B_n^R$) に分割しておく.(1) B_i^R と T をプロセッサ i で結合し，$T'_i = T \bowtie B_i^R$ を求める.(2) 各プロセッサの $s_i, T'^+_i, |T'_i|$ を一つのプロセッサに集め，$s = \sum_i s_i, T'^+ = \sum_i T'^+_i, |T'| = \sum_i |T'_i|$ から $gain$ を求める.

事例集合分割：(1) T を互いに素かつできる限りサイズの等しい n 個の集合 $T_1, \cdots, T_n$ ($T = T_1 \cup \cdots \cup T_n$) に分割する.(2) T_i と B^R をプロセッサ i で結合し，$T'_i = T_i \bowtie B^R$ を求める.(3) 各プロセッサの $s_i, T'^+_i, |T'_i|$ を一つのプロセッサに集め，$s = \sum_i s_i, T'^+ = \sum_i T'^+_i, |T'| = \sum_i |T'_i|$ から $gain$ を求める.

3. 実験結果

並列に結合演算を行う FOIL を Java を用いて実装し，分散メモリ型並列計算機 AP3000 の上で比較実験を行った．分割処理を行う親プロセッサ 1 台と分割された処理を行う子プロセッサを 15 台用意し，三つの並列化手法についてその実行時間と速度向上比を調べた．実験は，Michalski の東行き列車の問題 [4] を使用し，正負各 1,000 事例を用意して行った．

この結果を図 1 に示す．速度向上比が理想値を超えているのは，親プロセッサと子プロセッサの並列効果によるものである．実験に使用した問題では，ある一つの関係に属するタプル集合だけが他に比べてとても大きい．このため探索空間分割で大きな待ち時間が生じ，速度向上比を低くしている．

4. 考　察

実験の結果を踏まえて，次の三つの視点から **2.** で述べた並列化手法を比較する．

（a） プロセッサの待ち時間
（b） 繰返しごとの分割処理
（c） FOIL 以外の ILP アルゴリズムへの応用

探索空間分割：(a) については，追加されたリテラルの関係 R によって B^R のサイズが大きく異なり，結合演算に要する時間に差が出るので大きな待ち時間が生じる.(b) については，探索を一つ進めるごとに評価する節集合の分割処理が必要となる．ILP アルゴリズムは探索問題としてとらえることができるので (c) は可能である．

背景知識分割：(a) について，B^R を分割した集合のサイズの差はたかだか 1 なので，待ち時間はごくわずかである.(b) については，B^R をあらかじめ分割しておくので発生しない．FOIL の背景知識はタプル集合で表現されているため分割可能だが，背景知識が論理プログラムで与えられる場合にはタプル集合の結合演算ではなく導出計算により節を評価するため (c) は期待できない．

事例集合分割：(a) については，T を分割した集合のサイズの差もたかだか 1 なので，待ち時間はごくわずかである.(b) については，探索を一つ進めるごとに T の分割処理が必要となる．ILP アルゴリズムは事例集合をもつので (c) は可能である．

以上より，ILP アルゴリズムの並列化手法としては事例集合分割が適している．FOIL の並列化に限ると，(b) の点から背景知識分割が適している．

本論文で示した実装は，ILP アルゴリズムの特徴を利用した高速化技術を含んでいない．これらを考慮した並列化手法の検討は今後の課題である．

文　献

[1] H. Blockeel, L. De Raedt, N. Jacobs, and B. Demoen, “Scaling up inductive logic programming by learning from interpretations,” Report no.CW297, Katholieke Universiteit Leuven, 2000.
[2] H. Ohwada, H. Nishiyama, and F. Mizoguchi, “Concurrent execution of optimal hypothesis search for inverse entailment,” Proc. 10th Int’l Conf. on ILP,

pp.165–173, London, UK, July 2000.

[3] J.R. Quinlan, "Learning logical definitions from relations," Machine Learning, vol.5, no.3, pp.239–266, 1990.

[4] R.S. Michalski, "Pattern recognition as rule-guided inductive inference," IEEE Trans. Pattern Anal. & Mach. Intell., vol.2, no.4, pp.349–361, 1980.

（平成 13 年 12 月 10 日受付）