7B－3

帰納論理プログラミングシステムProgolの並列実装の設計*

尾崎 知伸† 村上 知子† 植野 研† 古川 康一†
慶應義塾大学 政策・メディア研究科‡

1 はじめに

帰納論理プログラミング (ILP;Inductive Logic Programming) は、一階述語論理上で帰納推論を行なうアプローチで、これまでの命題論理をベースとした帰納推論システムに較べ、表現力が拡張され、また背景知識を利用できるという特徴を持つ。しかし表現言語を一階述語論理に拡張したことで実行効率が落ち、大規模な問題に対して計算量の壁が問題となっており、実用化のための条件として、システムの高速化が望まれている。この問題の一つの解決策として、並列ILPシステムの構築が考えられる。

本稿では、S.Muggletonらによって開発されたILPシステムProgol[1]をターゲットとし、その並列性の検証、及び並列論理型言語KL1を実装言語とした並列設計に関して報告する。

2 並列性の検証

2.1 Progolの概要

Progolは、正事例、負事例、背景知識、及びモード宣言、タイプ情報を入力とし、背景知識と共に正事例を説明し、負事例を含まない仮説のうち最も簡潔なものを生成する。このタスクは、(1) 正事例集合の一つの要素と背景知識から最弱仮説を生成し、(2) 最弱仮説によって形成される候補仮説束内を探索し最適なものを選択する、(3) 得られた仮説によって説明できる正事例を正事例集合から取り除く、という処理を正事例集合が空になるまで繰り返すことで実現されている。

Progolは(2)において、$A^* - like$ アルゴリズムと呼ばれるトップダウン戦略を採用している。また、最適な仮説を選択する基準として、各候補仮説によって説明される正負事例の数、及び候補仮説のサイズを利用する。このため候補仮説が生成される度に、証明される正負事例の数を計算する。(この処理をカバーチェック計算と呼ぶ。)

2.2 システムのプロファイリング

表1は、リストを反転させる述語'reverse'を学習させる際、与える正負事例の大きさが実行速度に与える影響を示したものである。最弱仮説の生成に要する時間は、事例数に関係なくほぼ一定であるが、サーチ処理に要する時間は与える事例数に比例して増大している。また全体の処理時間に対するサーチ時間の割合は平均で99.8%、そのうちカバーチェック計算には平均87.2%の時間が費やされている。以上のことから、カバーチェック計算の効率的な並列化が、直接システム全体のパフォーマンスに影響を与えると言える。

事例数	最弱仮説の生成 (sec)	カバーチェック計算 (sec)	サーチ (sec)
400	0.032	6.74	7.84
800	0.050	15.97	18.51
1200	0.050	30.16	34.61
1600	0.032	48.12	54.85
2000	0.066	71.48	80.68

表1:Progolに与える事例数と実行速度

3 並列カバーチェックアルゴリズムの設計

計算の大部分を占めるカバーチェック計算を効率良く並列化することはシステム全体のパフォーマンスを決める大きな要因である。ここでは特に、カバーチェック計算の並列設計について考察する。(その他の部分の並列設計に関しては、[2, 3]を参照)

カバーチェック計算を行なうためには、Prologと同じ能力の定理証明器が必要となる。我々はトップダウン、ボトムアップの両アプローチから、いくつかの方式を検討した。

---

*A Design of Parallel Implementation of Inductive Logic Programming System Progol
†Tomonobu OZAKI, Tomoko MURAKAMI, Ken UENO, Koichi FURUKAWA
‡Graduate School of Media and Governance, Keio University

(1) 複数の Prolog プロセスの並列実行
(2) モデル生成型定理証明系:MGTP の利用
(3) 候補仮説の SQL 変換

(1)、(2) はトップダウン計算、(3) はボトムアップ計算である。

(1) は複数の Prolog プロセスをあらかじめ起動しておき、必要に応じて各プロセスに対してタスクを送信する方式である。この方式はオリジナルの Progol の処理をそのまま並列に行なうことになる。(Progol は事例を Prolog のゴールとして実行することで、カバーチェックを行なっている。) (2) は一階述語論理の並列ボトムアップ定理証明器である MGTP を利用することで、一度に全ての事例に対しカバーチェックを行なう方式である。この場合には、証明に無関係な節の生成の抑制、MGTP の条件である値域限定条件を満たす目的で、節のマジックセット変換が必要となる。この変換は、ボトムアップ計算でトップダウン計算をシミュレートするものであり、結果的にはトップダウン計算と等価であると言える。

(3) は ILP のデータベースからの知識獲得への応用を考えた際、特に有効なアプローチであると考えられる。このアプローチでは、仮説の本体部を SQL の条件にし、その条件文を満足するデータベース中の目標概念のインスタンス集合をボトムアップに求める。その際、前処理として無関係な節を排除し、カバーチェック計算の効率を高めることも考えられる。このようなボトムアップ的アプローチは、データベースと直結して考える際にトップダウン的アプローチより優れていると思われる。ここで任意の Prolog のプログラムは SQL に変換出来ないことを考慮する必要がある。SQL に変換される仮説は、Datalog (引き数に関数の現れないホーン節の部分集合) である必要があり、これは学習の際の一つの制限である。

以上トップダウンとボトムアップの両面からカバーチェックアルゴリズムの並列設計に関して考察を行なった。SQL を用いたボトムアップ計算は、並列性との親和性は高いが、Datalog しか扱えないという制限がある。どちらの戦略がより効果的か判断するためには、システム評価を行なう必要があり、今後の課題である。

4 Progol の利用環境

前章で述べたボトムアップアプローチを利用すると、データベースからユーザが与えたもの以外の事例を生成することができる (Figure 1)。このことを利用すると、デバッグ機能を含んだシステムの利用環境の向上が期待できる。

新たに発見された事例を、正事例として取り込むかどうかを、ユーザに問合わせることで、対話的かつ漸増的に正事例を増加させることが可能となる。

また、必要な背景知識の欠落により、本来カバーされるべき正事例がカバーされない場合がある。この場合、背景知識の欠落によりカバーされなかった事例と説明に必要な背景知識を提示し、欠落した背景知識をユーザに問い合わせることを検討している。

これからの課題として、これらの機能をシステムの利用環境として実装し、データベースからの知識獲得に応用していく予定である。

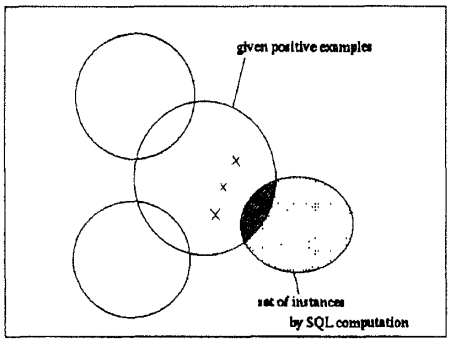

Figure 1

5 おわりに

本稿では Progol をターゲットとした、並列性の検証及び設計について報告した。また Progol の利用環境に関しても考察を行なった。この設計に基づく並列実装及び評価は今後の課題である。

参考文献

[1] S. Muggleton: Inverse entailment and Progol, New Generation Computing, Vol.13, pp.245-286 (1995).

[2] 藤田博, 古川康一, 八木直樹: MGTP による PROGOL の最弱仮説の生成, 人工知能学会第 10 回全国大会論文集,pp53-54 (1996).

[3] Hiroshi FUJITA, Naoki YAGI, Tomonobu OZAKI, Koichi FURUKAWA: A New Design and Implementation of Progol by Bottom-up Computation, Proc. of ILP'96, (1996)