PARCAR：コストに基づく並列仮説推論システム

加藤昇平[†] 世木博久[††] 伊藤英則[††]

本稿では，コストに基づく仮説推論においてその最適解探索法の並列化を提案し，並列仮説推論システムを実現する．並列ヒューリスティック探索が持つ探索制御技術を仮説推論の推論機構に導入し，与えられた観測を説明する最小コストの仮説集合を効率的に求めるものである．並列仮説推論システムPARCAR（PARallel Cost-based Abductive Reasoning system）をMIMD型分散メモリ並列計算機である富士通AP1000上に実装し，その実験結果についても報告する．

PARCAR: A Parallel Cost-based Abductive Reasoning System

Shohei Kato,[†] Hirohisa Seki[††] and Hidenori Itoh[††]

We propose an efficient parallel first-order cost-based abductive reasoning system, which can find a minimal-cost explanation of a given observation. In this paper, we introduce a search control technique of parallel best-first search into abductive reasoning mechanism. We also implement a PARallel Cost-based Abductive Reasoning system, PARCAR, on an MIMD distributed memory parallel computer, Fujitsu AP1000, and show some performance results.

1. はじめに

知識ベースシステムは，AIの基礎的研究テーマであると同時にAIの実用化や知識情報処理システム構築のための要素技術として不可欠な研究課題である．また，実用化に耐える規模の知識ベースシステムが要求する計算能力にこたえるためには並列処理が欠かせないものと考える．本研究では，知識ベースシステムの推論機構として高次推論の一形式である仮説推論に着目し，その並列化について検討する．

仮説推論は，与えられた観測を説明するために，一時的に成り立つ知識をとりあえず正しいと仮定したり，足りない知識を仮説で補うなど，柔軟で知的な推論処理である[9),11)]．しかしながら，仮説推論システム実現上の問題点として，仮説推論の計算量がきわめて大きいことが知られており[11)]，仮説推論の探索空間を絞り込む工夫について多くの研究結果が報告されている．たとえば，文献8)，13)，15)などにおいて，演繹データベースの分野における問合せ処理のさまざまな最適化技術が仮説推論に応用され効率の良い仮説推論が提案されている．

仮説推論では一般に，観測を説明する仮説集合は複数存在する．しかしながら，診断・計画問題などにおいて見られるように，ユーザが要求する解はすべての説明ではなく，むしろ最も好ましい説明であることが多い．

たとえば，計画問題においては，与えられた目標を達成するすべての手順よりも，それらの中で最も安価（あるいは最速）な手順がしばしば要求される．このような要求に対しては，仮説に好ましさの基準（確率，コストなど）を与える手法[2),10)]が報告されている．そこで文献14)では，A*アルゴリズムを仮説推論に応用することにより，最も好ましい説明を求める仮説推論が提案されている．

本稿では，これらの工夫に加えて，仮説推論の探索空間を複数のプロセッサへ分散し推論処理を並列化することにより，仮説推論の高速化を実現する．仮説推論の推論処理技術に並列ヒューリスティック探索が持つ探索制御技術を導入し，与えられた観測を説明する最小コストの仮説集合を効率的に求めることにより最適な説明を得る並列仮説推論を提案する．一般的にA*などによる推論処理の高速化はヒューリスティック評価関数の精度に大きく依存する．本稿で提案する推論処理の並列化により，同評価関数の精度が悪い場合で

† 豊田工業高等専門学校電気・電子システム工学科
Department of Electrical and Electronic Engineering, Toyota National College of Technology
†† 名古屋工業大学知能情報システム学科
Department of Intelligence and Computer Science, Nagoya Institute of Technology

も高速な推論処理が実現される．

本稿で提案する並列仮説推論システム PARCAR（PARallel Cost-based Abductive Reasoning system）を MIMD 型分散メモリ並列計算機 AP1000 上に実装し，その実験結果についても報告する．

2. コストに基づく仮説推論

本稿では，仮説の選択の基準として知識ベースに含まれる各仮説に重み（コスト）が与えられている場合を対象に，与えられた観測に対して最適な説明を求める仮説推論を考える．各仮説に与えられる重みは正の数とし，仮説集合のコストは集合に含まれる仮説の重みの和で与えられるものとする．説明が最適であるとは説明を表す仮説集合のコストが最小であることとする．

定義 2.1 F をホーン節集合（「事実」と呼ぶ），H を基底単位節の集合（「仮説集合」と呼ぶ）とする．また，O を存在束縛されたアトムの連言（「観測」，または単に「ゴール」と呼ぶ）とする． このとき，O の $F \cup H$ による最適な説明とは，以下の条件を満足するような，H の要素から成る集合 E を求めることである．

$F \cup E \vdash O$ （$F \cup E$ から O が証明される）(AR1)

$F \cup E \nvdash$ false（$F \cup E$ は無矛盾である） (AR2)

$cost(E) \leq cost(D)$ for all $D : F \cup D \vdash O, F \cup D \nvdash$ false （条件 AR1，AR2を満たす集合の中で E のコストが最小） (AR3)

$cost(H)$ は仮説集合 H に含まれる仮説のコストの和を表すとする．ここで，F はつねに成り立つ知識として扱われる．一方で，H の部分集合は F と矛盾する可能性がある．また，F と H の和集合をプログラムと呼び，P で表記する． ■

本稿では仮説の形式を基底単位節に限定しているが，以下のようなホーン節の変形[12]により，ホーン節の形式をとった仮説にも対応できる．

仮説：$b \leftarrow a_1, a_2, \cdots, a_n.$ コスト：c

を以下の事実と仮説へ変形する

事実：$b \leftarrow a_1, a_2, \cdots, a_n, e.$

仮説：e コスト：c

定義 2.2 F の中に存在する負節を「一貫性制約節」（あるいは単に「制約式」）と呼ぶ．制約節は，論理式 false $\leftarrow A_1, \cdots, A_n$ で表現される．ただし，A_k $(1 \leq k \leq n;\ n \geq 1)$ はアトムであり，false は「矛盾」を表す． ■

例 2.1 **表 1** に示す簡単な例題知識ベース P_{ex} を考える．P_{ex} に対して，問合せ "$\leftarrow p(X,Y)$" を与える．

表 1 例題知識ベース P_{ex}
Table 1 An example P_{ex}.

事実	仮説	コスト	仮説	コスト
p(X,Y) ← a(X),c(Y).	c(1).	5	c(2).	6
p(X,Y) ← b(X),d(Y).	c(3).	7		
a(X) ← e(X).	b(1).	2	b(2).	3
e(1).	b(3).	2		
false ← b(X),d(X).	d(1).	2	d(2).	4

同表の知識ベースが事実と仮説の和集合 $F \cup H$ であり，問合せ $\leftarrow p(X,Y)$ が観測 O である．

図 1 に例 2.1 に対する SLD 反駁[6]を用いた仮説推論の実行例を示す．同図においてゴール節に現れる M なるオペレータが付いたアトム "MA" は，サブゴール $\leftarrow A_0$（ここでアトム A は A_0 のある代入例）を解く際に，仮説を入力節として導出を行ったこと（すなわち，A に対して仮説が立てられたこと）を示すためのマークである．また，ゴール内のアトムの選択規則は最左優先とする．

ここで，図 1 の←falseに対する SLD 反駁木から，成功葉に現れる仮説集合$\{b(1), d(1)\}$または$\{b(2), d(2)\}$を仮定することは知識ベースに対して矛盾を起こすことが分かる（これらの仮説集合を矛盾仮説と呼ぶ）．また，同図の$\leftarrow p(X,Y)$に対する SLD 反駁木より，右から 2 番目の成功葉から得られる仮説集合$\{b(3), d(1)\}$のコストが反駁木の任意の成功葉から得られる仮説集合のコストにおいて最小であるので，これを説明としてゴール $\leftarrow p(X,Y)$ の最適解$\{X = 3, Y = 1\}$が得られる． ■

しかしながら，1 つの最適解を得るために全解を求めるのは，無駄な探索空間が大きくきわめて非効率である．そこで，与えられた観測に対する最適な説明を効率的に求めるために，本仮説推論が対象とする探索空間の評価関数を以下のように定義する．

定義 2.3 P をプログラム，O を与えられた観測とする．また，P におけるゴール$\leftarrow O$に対する SLD 反駁木（$P \cup \{\leftarrow O\}$ の SLD 反駁木と呼ぶ）における任意のゴール節 $g = \leftarrow \mathsf{M}L_1, \cdots, \mathsf{M}L_i, A_{i+1}, \cdots, A_n$ について，$\hat{h}(g)$ を g から成功葉に至る最適導出におけるコストの予測値，$h(g)$ を実際のコストの値とする．このとき，最適解探索における g の評価値 $f(g)$ を以下のように定義する．

$$f(g) = cost(\{L_1, \cdots, L_i\}) + \hat{h}(g)$$

$\hat{h}(g)$ は以下の条件を満足するような予測値とする．

$$0 \leq \hat{h}(g) \leq h(g)$$

$$h(g) = cost(\bigcup_{j=i+1}^{n} A_j \text{の最適な説明})$$

■

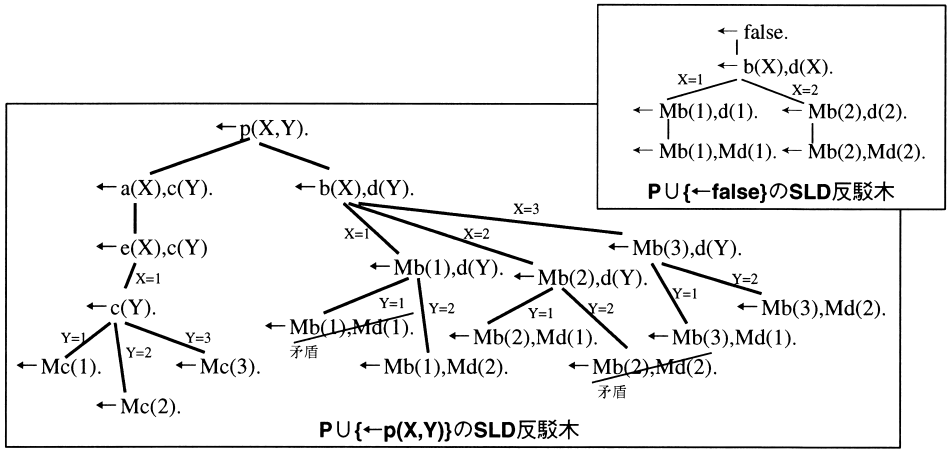

図 1 仮説推論の実行例
Fig. 1 An example of abductive reasoning for P_{ex}.

ここで，$cost(\{L_1, \cdots, L_i\})$ は反駁木の根から g に至る導出におけるコストの値を示している．また，本稿では，A_j の説明が存在しないときには，$h(g) = \infty$ とする．

仮説推論の探索空間に対してこのような評価関数が与えられることによって，最適な説明を求める問題は最適解探索の問題に帰着される．そこで文献 14) では，SLD 反駁に対して A* アルゴリズムが持つ探索制御技術を導入することにより効率的な仮説推論システムが提案されている．文献 14) で実現した仮説推論の探索空間の絞り込みに加えて，本稿では，その探索空間を複数のプロセッサに分散することによる仮説推論システムの並列化について報告する．なお，本稿では，知識ベースに現れる制約式については←falseに対する反駁☆を行い，矛盾仮説の集合をあらかじめ求めておくことにする．

3. 並列仮説推論システム PARCAR

本章では，並列仮説推論システム PARCAR のアルゴリズムについて説明する．並列ヒューリスティック探索が持つ探索制御機構を仮説推論の推論機構に導入し，複数のプロセッサが同期・通信を通じて並列に推論を行うことにより最適な説明を効率的に求めるものである．仮説推論のノード展開は述語のユニファイや変数の束縛情報の伝搬などをともなう SLD 導出による．したがってノード展開にかかる計算量は大きく，生成される子ノードの数によってその計算量は大きく異なる．そこで PARCAR では，各プロセッサで 1 回の反復で行う推論の処理量は展開されるノード数ではなく，生成されるノード数に基づくものと考え，全プロセッサ間でそのノード数が一定になるように負荷分散を行っている．また，仮説推論では展開されるノードは述語論理ホーン節で表現される．したがってノードのデータ構造は複雑でありノード分散にかかる通信コストも大きくなる．そこで PARCAR では各プロセッサが 1 反復に処理する処理量をある程度大きくし，プロセッサ間の通信回数を抑え，通信コストを抑えている．

図 2 に PARCAR の処理のながれを示す．知識ベースおよび観測が与えられると PARCAR は「進行（progress）フェイズ」と「確認（confirmation）フェイズ」の 2 段階の状態を経て最適な説明を生成する．

「進行フェイズ」では，すべてのプロセッサは同期をとりながら「ゴール節展開」「サブゴール節分散」「サブゴール節受信」の 3 つの手続きを反復する.「ゴール節展開」手続きにおいて各プロセッサは，各々が持つ探索空間の未展開ノード（**図 3** 中○印のノード）をそれらの評価値に関する最良優先で展開し探索木を生成する.「サブゴール節分散」手続きにおいて各プロセッサは,「ゴール節展開」手続きにより生成された探索木の葉にあたるノード（図 3 中破線で囲まれた部分における○印のノード）を全プロセッサに分散する．そして,「サブゴール節受信」手続きにおいて各プロセッサは，全プロセッサから届いたノードを受け取り未展開ノードの集合（図 3 中 $OPEN_i^j$）を更新する.「進行フェイズ」において，あるプロセッサが与えられた観測の説明（解候補）を生成すると，すべてのプロセッサは「確認フェイズ」に移行する．

「確認フェイズ」では，すべてのプロセッサは各々

☆ PARCAR を用いてこの反駁を行うことも可能だが，その処理自体は，知識ベースの規模が大きい場合には仮説推論処理と比較して無視できるほど軽い．

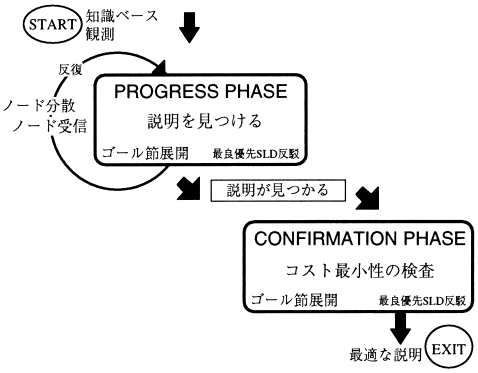

図 2 PARCAR の処理の流れ
Fig. 2 Two phases.

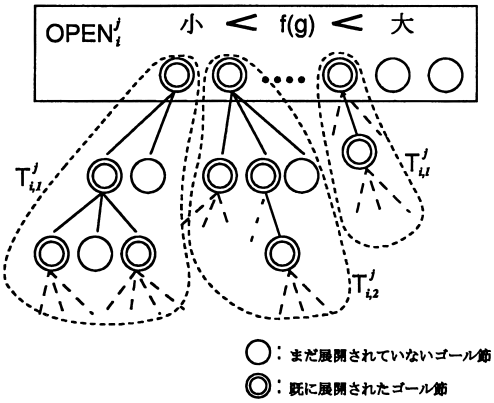

図 3 プロセッサ j で生成される SLD 反駁木の模式図（反復 i 時）
Fig. 3 A model of a Sub-SLD-tree constructed by processor j at i-th iteration.

が持つ探索空間の未展開ノードに対して「ゴール節展開」手続きを実行し，解候補のコスト最小性を検査する．同手続きにおいて新たな説明が得られた場合には，観測の説明は更新される．

図 4 に PARCAR のアルゴリズムを示す．同図において，n はプロセッサの台数，$OPEN_i^j$, $CLOSED_i^j$, $Gmin^j$, SG^j, RS^j および ANS^j はプロセッサ j ($1 \leq j \leq n$) で扱われるゴール節の集合を表す．$OPEN_i^j$, $CLOSED_i^j$, SG^j は，i 回目の反復において，プロセッサ j によってまだ展開されていないゴール節（図 3 中○印のノード）の集合，すでに展開されたゴール節（図 3 中◎印のノード）の集合，および，形成された部分 SLD 反駁木（図 3 中破線で囲われた部分）の葉に現れるゴール節の集合をそれぞれ示している．また ANS^j はプロセッサ j において導出された最適な説明の候補を表すゴール節の集合を示す．なお，これらの集合は要素であるゴール節 g の評価関数 $f(g)$ に基づいた順序つき集合として計算機上に実装される．

直観的には，プロセッサ j における i 回目の反復で「ゴール節展開」「サブゴール節分散」「サブゴール節受信」の 3 手続きは次のような動作をとる．$T_{i,l(1\leq l)}^j$ をプロセッサ j で形成される部分 SLD 反駁木☆とし，$|T_{i,l}^j|$ を $T_{i,l}^j$ に含まれるゴール節の数とする．

ゴール節展開手続き 同手続きがとるフェイズに基づき以下のいずれかの手続きを実行する．

進行（*progress*）フェイズ

$OPEN_i^j$ に含まれるゴール節を根とするような部分 SLD 反駁木 $T_{i,l}^j$ をゴール節の評価値に対する最良優先で生成する（図 3 参照）．$T_{i,l}^j$ の生成は $\sum_l |T_{i,l}^j| < K$ を満たす間実行される．K は推論開始時にユーザから与えられるパラメータである．ただし， 解が得られた，あるいは，展開するゴール節が存在しない場合にはただちに手続きを終了する．

確認（*confirmation*）フェイズ

$OPEN_i^j$ に含まれるゴール節を根とするような部分 SLD 反駁木 $T_{i,l}^j$ をゴール節の評価値に対する最良優先で生成する．t を進行フェイズにおいて求められた最適な説明の候補のコストとすると，$T_{i,l}^j$ の生成は，$T_{i,l}^j$ に現れるすべての葉のゴール節 g が $f(g) \geq t$ となる，あるいは，t よりも評価値が小さい新しい解が得られるまで実行される．PARCAR はこのフェイズを経て成功裏に終了する．

サブゴール節分散手続き ゴール展開手続きにおいて最適解の候補が得られたかどうかを調べ，得られた場合には，そのコストおよび確認フェイズへの移行要求を全プロセッサに放送する．そして，同手続きにおいて生成された部分反駁木 $T_{i,l}^j$ の葉のゴール節をその評価値が小さいものから順にゴール節の数が均一となるように全プロセッサに分散する．このとき，あるプロセッサに評価値の小さいゴール節が集中しないようにプロセッサ j で生成されたゴール節でその評価値が r 番目に小さいものはプロセッサ $(r+j) \bmod n$ へ送信する．

サブゴール節受信手続き あるプロセッサから確認フェイズへの移行要求が届いたらフェイズを移行し，*shifted* の値を true にする．このとき，付随して届く最適解の候補のコストを確認フェイズでの探索終了条件となる閾値 t に代入する．なお，複数のプロセッサから同様のコストが届いた場合にはそ

☆ 図 3 に示すように複数の SLD 反駁木が生成される．l は識別子．

並列仮説推論システム PARCAR
入力: プログラム P, 観測 O
出力: 解: (解代入例, 最適な説明)（成功裏）または false（失敗裏）

```
1 begin
2   OPEN_0^1 := {←O} ; i := 0 ;  phase^j := progress ; t := ∞ ;
3   OPEN_0^{j(1<j≤n)} := φ ;  CLOSED_0^{j(1≤j≤n)} := φ ; ANS^{j(1≤j≤n)} := φ ;
4   repeat
5     OPEN' := OPEN_i^j ; CLOSED' := φ ; SG^j := φ ;  escape := false ; shifted := false ;
6     repeat                                              % ゴール節展開手続き
7       Gmin^j := {^∀g ∈ OPEN' ∪ SG^j | for all g' ∈ OPEN' ∪ SG^j : f(g)≤f(g')} ;
8       Gmin^j に対して Pにおける 1 段階の SLD 導出で得られるサブゴールをすべて求め
        これらの集合を RS^j とする ;
9       RS^j について無矛盾性の検査†を行い Pと矛盾するゴールを RS^j から削除する ;
10      ANS^j := {^∀g∈RS | gは成功葉, かつ, 成功葉のコスト中で f(g) が最小 } ;
11      SG^j := SG^j∪RS^j ;
12      CLOSED' := CLOSED' ∪ {g} ;
13      OPEN' := OPEN' \ Gmin^j ; SG^j := SG^j \ Gmin^j ;
14      if (ANS^j ≠ φ)      then escape := true ;
15      if ( phase^j = progress and (|SG^j| + |CLOSED'|≥K or OPEN' ∪ SG^j = φ) )
                            then escape := true ;
16      if ( phase^j = confirmation and (^∀g ∈ OPEN' ∪ SG^j | f(g) > t or ^∃g ∈ ANS^j | f(g) < t )
                            then escape := true ;
17    until (escape)
18    OPEN_{i+1}^j := OPEN' ;  CLOSED_{i+1}^j := CLOSED_i^j ∪ CLOSED' ;
19    if (phase^j = progress) then
20      begin
21        distribution_goals(ANS^j, SG^j) ;                 % サブゴール節分散手続き
22        receiption_goals(OPEN_{i+1}^j, phase^j, shifted) ;  % サブゴール節受信手続き
23      end
24    i := i + 1 ;
25  until ( (phase^j = confirmation and not shifted) or (OPEN_i^{j(1≤j≤n)} = φ) )
26  ANS := {^∀g∈ANS ∪ ANS^{j(1≤j≤n)} | for all g'∈ANS ∪ ANS^{j(1≤j≤n)} : f(g)≤f(g')}
27  if (^∃←ML_1 ··· ML_m ∈ ANS) then return （解代入例,{L_1···L_m}）
28  else return false
29 end.
```

† サブゴールに現れる仮説が矛盾仮説を含むかどうかで検査する.

図 4 並列仮説推論システム **PARCAR** のアルゴリズム
Fig. 4 The **PARCAR** algorithm.

の最小値を t に代入する．全プロセッサから送信されたゴール節を $OPEN_{i+1}^j$ に追加する．このとき，$\bigcup_j^n OPEN_{i+1}^j = \phi$ となれば PARCAR は失敗裏に終了する．

本アルゴリズムでは，各プロセッサは上記 3 手続きにおいて直接同期をとるためのフラグを持っておらず，これらの手続きは非同期に処理される．しかしながら，「サブゴール節受信」手続きは全プロセッサからデータを受信するまで処理が終了しない仕組みになっている．これにより 3 手続きの反復回数については全プロセッサ間で同期が実現されている．

任意の有限な SLD 反駁木において，すべてのゴール節 g に対し条件 $\hat{h}(g) \leq h(g)$ が成立するとき，本並列アルゴリズムは実行可能であり，解が存在する場合には必ず最適解を検出する．

4. 関連研究

演繹推論の並列化に関しては，代表的な演繹推論機構である Prolog の OR-並列化について多くの研究報告がされている（文献 1), 7) など）．Prolog が形成する探索空間が深さ優先の SLD 反駁であるのに対して，本稿が対象とする仮説推論ではコスト最小の説明が要求されるため，幅優先で反駁木が形成される．このため探索の並列性はより高く，探索空間の分散が容易に行える．各プロセッサは割り当てられた部分空間に対して最良優先探索を行う．

文献 3) では A* アルゴリズムを並列化した PIA* アルゴリズムが提案されている．PIA* では i 回目の反復で各プロセッサ j は優先度付き待ち行列 WL_i^j（work list）（PARCAR における $OPEN_i^j$ に相当）に含ま

表 2 例題知識ベース ($n = 1$ の場合)
Table 2 An example knowledge-base (incase $n = 1$).

事実
val(Node2,V) ← conn(Node1,Node2),val(Node1,V).
val(out(D),V) ← dev(D,and),ok(D),val(in(1,D),A),val(in(2,D),B),and(A,B,V).
val(out(D),V) ← dev(D,or),ok(D),val(in(1,D),A),val(in(2,D),B),or(A,B,V).
val(out(D),V) ← dev(D,xor),ok(D),val(in(1,D),A),val(in(2,D),B),xor(A,B,V).
val(out(D),1) ← dev(D,Ano),stuck_on(D).
val(out(D),0) ← dev(D,Ano),stuck_off(D).
conn(in(1,x1),in(1,g1x)). conn(in(1,y1),in(2,g1x)). conn(in(1,x1),in(1,g11)). conn(in(1,y1),in(2,g11)).
conn(out(g0c),in(2,g12)). conn(out(g0c),in(2,g1z)). conn(out(g1x),in(1,g1z)). conn(out(g1x),in(1,g12)).
conn(out(g11),in(2,g1c)). conn(out(g12),in(1,g1c)).
and(0,0,0). and(0,1,0). and(1,0,0). and(1,1,1). xor(0,0,0). xor(0,1,1). xor(1,0,1). xor(1,1,0).
or(0,0,0). or(0,1,1). or(1,0,1). or(1,1,1). val(in(1,x1),0). val(in(1,y1),1). val(out(g0c),0).
dev(g1x,xor). dev(g1z,xor). dev(g11,and). dev(g12,and). dev(g1c,or). dev(g0c,or).
false ← ok(D),stuck_on(D). false ← ok(D),stuck_off(D). false ← stuck_on(D),stuck_off(D).

仮説: コスト
ok(g1x). : 0.500 ok(g1z). : 0.500 ok(g11). : 0.500 ok(g12). : 0.500 ok(g1c). : 0.500 ok(g0c). : 0.500
stuck_on(g1x). : 0.170 stuck_on(g1z). : 0.170 stuck_on(g11). : 0.190 stuck_on(g12). : 0.190
stuck_on(g1c). : 0.220 stuck_on(g0c). : 0.110 stuck_off(g1x). : 0.330 stuck_off(g1z). : 0.330
stuck_off(g11). : 0.310 stuck_off(g12). : 0.310 stuck_off(g1c). : 0.280 stuck_off(g0c). : 0.390

観測 ← val(out(g1z),1),val(out(g0c),1).

れるノードの中で評価値が閾値 t^i 以下のノードに対して 1 段階の導出を行う．ここで t^i の値はすべてのプロセッサが持つ $WL_i^{j(1\leq j\leq n)}$ に含まれるノードの評価値の最小値をとるので A* アルゴリズムを単純に並列化したものとなっている．また，各プロセッサで展開されたノードの分散は PARCAR と同様の方式をとっている．

各プロセッサによって展開されるノードの数は，PIA* のノード展開の手続きより均等になる．しかしながら，一般的にはノードによって展開時に生成される子ノードの数は異なる．たとえば表 1 の例題に対して PIA* では 2 回目の反復においてプロセッサ 1 はゴール節←$a(X), c(Y)$，プロセッサ 2 はゴール節←$b(X), d(Y)$を展開する（図 1 の←$p(X)$に対する SLD 反駁木の深さ 2〜3 の展開）．プロセッサ 1 で生成される子ノード数は 1 であるのに対し，プロセッサ 2 は 3 個の子ノードを生成する．本仮説推論においては，子ノード生成は述語のユニファイ，変数の束縛情報の伝播などの処理をともないその計算量は無視できないものとなり，生成される子ノードの数の差☆1が PIA* の負荷分散を結果的に不均衡にする要因となる．

プロセッサの処理の単位を生成される平均ノード数とした場合の各プロセッサの粒度は，PIA* では細かい☆2．したがってプロセッサ間の通信回数が増大し，台数を増やした場合に通信コストがボトルネックとなり得る． 一方，PARCAR では粒度はパラメータ K にほぼ等しく，K の値を調節することによってそのばらつきを抑えつつ，通信コストを軽減している．

5. 実験結果

本稿で提案した並列化の有効性を確認するために，推論時間の比較実験を行った．例題としては，論理回路の故障診断問題を用いた．回路は n-ビット全加算器を対象にした．$n = 1$ の場合の例題知識ベースを**表 2** に示す．回路を構成する素子は「正常」「開放」「短絡」の 3 つの状態をとるものとし，これを仮説として表現した．また，各仮説のコストは，各素子が上記の状態となる確率 $\mathcal{P}$ を考え，$-\log_e \mathcal{P}$ で与えた☆3．n をパラメータとして n-ビット全加算器回路の規模を変化☆4させて実験を行った．実験は富士通 AP1000 上で C 言語を用いて行った．AP1000 は MIMD 型分散メモリアーキテクチャの並列計算機である．

実験結果を**図 5** に示す．PARCAR 実行時に与える

☆1 実験で扱った例題（5 章参照）では 1 個のゴール節展開につき最大 6 となる．

☆2 5 章で行った実験では PIA* の粒度は 2.1 となった．

☆3 確率の積 $\Pi_i P_i$ が正の数の和 $\Sigma_i(-\log_e P_i)$ で表現でき，確率最大の説明を求める問題がコスト最小の説明を求める問題に帰着される．

☆4 この問題を表現する知識ベースの規模は n に比例し，事実の知識は $17n + 23$ の述語論理ホーン節，仮説は $15n + 3$ の単位節から成る．

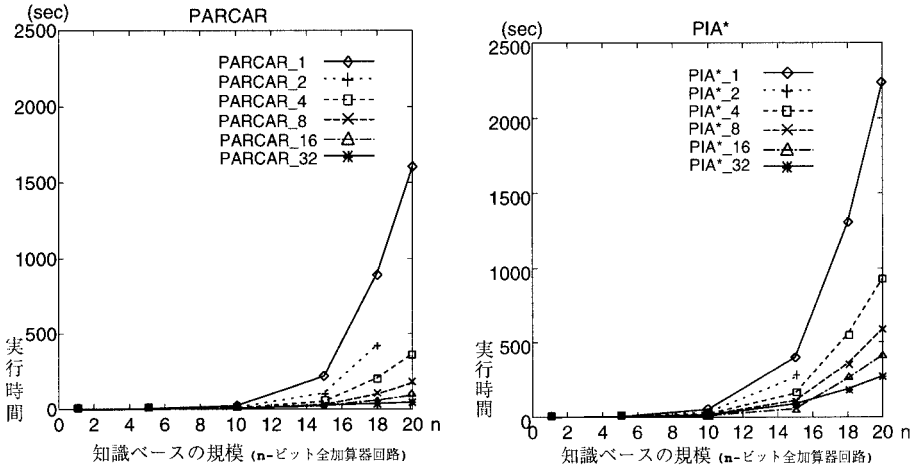

図 5 実験結果
Fig. 5 Experimental results.

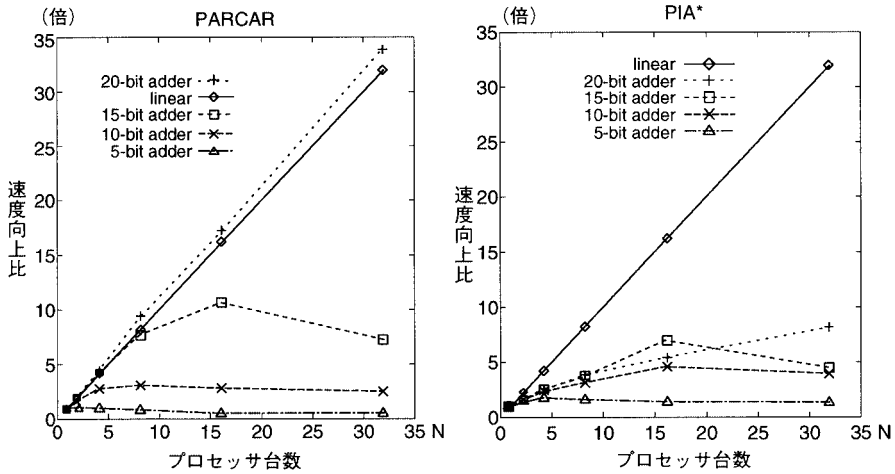

図 6 速度向上比
Fig. 6 Speedups.

パラメータ K の値は 50 とした．両手法とも，探索木に現れるゴール節 g のコストの予測値（定義 2.3 参照）は $\hat{h}(g) = 0$ とした．図 5 左が PARCAR を用いた結果，右が PIA* を用いた手法の結果をそれぞれ示している．同図中プロセッサ数 2 におけるデータの欠損は，メモリ容量の不足☆により推論が実行不可能であったことを意味している．なお，プロセッサ数 1 の実験に関しては同性能の CPU を持つメモリ容量の大きな逐次計算機を用いて実験を行った．知識ベースの規模が増大するほど，PARCAR が PIA* を用いた手法に比べて優れていることが分かる．特に問題規模 $n = 20$，プロセッサ台数 32 において本システムは PIA* を用いた手法の約 5.8 倍の推論速度が達成されている．**図 6** に両手法の速度向上比を示す．PIA* を用いた手法では並列化による台数効果は問題規模の大小にかかわらずプロセッサ数の少ない時点から線形より大きく外れるが，PARCAR では問題規模が大きくなるにつれてプロセッサ台数を増やした場合でもほぼ線形の台数効果が得られているのが分かる．

次に，問題規模 $n = 20$ における実験結果の詳細を**表 3** に示す．同表にて，$|CLOSED|$ は両アルゴリズムが最適解を見つけるまでに展開したゴール節の数を示しており，探索空間の大きさの程度を表している．$\overline{Aber}(T_{exp})$ は各プロセッサでのゴール節展開手続きの処理時間の平均時間に対するばらつきの度合いを示す

☆ 実験で用いた並列計算機 AP1000 は，各プロセッサごとに 16 MB のメモリを保有しており，システム全体として使用可能なメモリ容量はプロセッサ台数に比例する．

表 3 問題規模 $n = 20$ における実験結果の詳細
Table 3 The detail of the results on the problem with its size $n = 20$.

		PARCAR				PIA*			
PE 数		32	16	8	4	32	16	8	4
実行時間	(sec)	47	94	174	366	275	416	593	931
反復回数		15	40	62	120	571	733	925	1137
$\|CLOSED\|$		24508	23653	23275	23437	22473	22978	23046	23121
$\overline{Aber}(T_{exp})$	(%)	55.8	12.6	7.7	8.3	71.4	57.3	48.5	45.5
$\overline{T}_{comm}$	(sec)	1.089	1.241	1.457	1.740	0.588	0.552	0.592	0.774

値であり，両アルゴリズムについて測定データを以下のように処理した結果である．T_{exp} を 1 回の反復におけるゴール節展開手続きの処理時間とし，$\overline{T}_{exp}$ を全プロセッサでの T_{exp} の平均時間とする．T_{exp} について各反復ごとに全プロセッサに対する標準偏差 $Dev(T_{exp})$ を求め，$\overline{T}_{exp}$ に対する比 $Aber(T_{exp}) = \frac{Dev(T_{exp})}{\overline{T}_{exp}} \times 100$ を調べる．そして，$Aber(T_{exp})$ の総反復での平均を求めたものである．また，表 3 最下段の$\overline{T}_{comm}$は 1 回の反復におけるプロセッサ間通信時間☆の全プロセッサにおける平均を示している．両アルゴリズムとも，全プロセッサにゴール節がいきわたった以降の反復について測定を行った．

表 3 中 $|CLOSED|$ より，PARCAR と PIA* はともに同程度の大きさの探索空間を費やしており，両者の実行時間の優劣はアルゴリズムの効率に起因していると考えられる．そこで，まず両者のプロセッサ間通信の処理時間について調べた．表 3 の結果より，1 回の反復において，PARCAR は PIA* の 1.9〜2.3 倍の通信時間を費やしているが，反復回数では 1/38〜1/10 の回数に抑えられている．各プロセッサでのプロセッサ間通信の平均時間は $\overline{T}_{comm} \times$ 反復回数 であることから，PARCAR は各プロセッサの処理粒度を適度に大きくすることによりプロセッサ間の通信回数を抑え，プロセッサ間通信のコストを軽減していることが分かる．

また，アルゴリズムの並列性を向上させるには，各反復における各プロセッサのゴール節展開手続きの処理時間のばらつきを抑え，各プロセッサの負荷をできるだけ均等にすることが重要である．表 3 中 $\overline{Aber}(T_{exp})$ より，たとえばプロセッサ数 8 においては，PARCAR ではゴール節展開手続きの処理時間が $\overline{T}_{exp} \pm 7.7\%$ 以内にほぼ収まっているのに対して，PIA* では同手続きの処理時間のばらつきが $\overline{T}_{exp}$ の 45% を超える結果となっている．したがって，各プロセッサの負荷分散においても，PARCAR は PIA* よりも優れていることが分かる．

本章では，論理回路の故障診断問題を例題とした推論時間の比較実験をもとに両手法を評価した．定量的な解析による両手法の評価においても PARCAR の有効性が報告されている．これについては文献 4) を参照されたい．

6. おわりに

コストに基づく仮説推論においてその最適解探索法の並列化を提案し，並列仮説推論システム PARCAR を実現した．本稿で提案した仮説推論システムを並列計算機 AP1000 上に実装し，有効性を確認した．

本稿では，仮説推論における最適解探索の並列化について探索空間の分散方法を中心に並列仮説推論システム PARCAR を説明した．本稿では推論時における $h(g)$ の下限予測値を $\hat{h}(g) = 0$ とした実験結果について報告したが，$\hat{h}(g)$ の事前解析方法および $\hat{h}(g)$ の精度向上による推論制御の効果については文献 14) を参照されたい．

最後に，一般的な OR 並列探索における評価値 $h(g)$ の下限予測値と並列効果について述べる．下限予測値 $\hat{h}(g)$ の精度向上と並列効果はドレードオフの関係にある．すなわち，$\hat{h}(g)$ が理想的な状態 ($\hat{h}(g) = h(g)$) に近づくにつれて探索空間の or-分岐数は減少し並列化の効果は下がる．仮説推論においては，与えられる知識ベースの規模や複雑さによっては，文献 14) などによる $\hat{h}(g)$ の精度向上は困難である．本稿で提案した並列化は，このような問題に対して特に有効であると考える．

謝辞 本研究は，一部，(財) 堀情報科学振興財団の助成により行われた．

参考文献

1) Ali, K.A.M. and Karlsson, R.: The Muse Or-Parallel Prolog Model and its Performance, *Proc. North American Conf. on Logic Programming*, pp.757–776, The MIT Press (1990).
2) Charniak, E. and Shimony, S.E.: Cost-based

☆ サブゴール節分散およびサブゴール節受信手続きの処理時間の和である．

abduction and MAP explanation, *Artificial Intelligence*, Vol.66, pp.345–374 (1994).
3) Huang, S. and Davis, L.S.: Parallel Iterative A^* Search: An Admissible Distributed Heuristic Search Algorithm, *Proc. IJCAI-89*, pp.23–29 (1989).
4) Kato, S., Seki, H. and Itoh, H.: Parallel Cost-Based Abductive Reasoning for Distributed Memory Systems, *N. Foo and R. Goebel eds. PRICAI'96: Topics in Artificial Intelligence*, Lecture Notes in Artificial Intelligence, Vol.1114, Cairns, pp.300–311, Springer-Verlag (1996).
5) Kato, S., Seki, H. and Itoh, H.: A Parallel Implementation of Cost-Based Abductive Reasoning, *Proc. 2nd Intl. Symp. on Parallel Symbolic Computation* (*PASCO'97*), Maui, Hawaii, pp.111–118, ACM press (1997).
6) Lloyd, J.W.: *Foundations of Logic Programming*, Springer (1984). 2nd, extended edition (1987).
7) Lusk, E., Warren, D.H.D., Haridi, S., et al.: The Aurora or-parallel Prolog system, *New Generation Computing*, Vol.7(2,3), pp.243–271 (1990).
8) Ohta, Y. and Inoue, K.: Incorporating Top-Down Information into Bottom-Up Hypothetical Reasoning, *New Generation Computing*, Vol.11, pp.401–421 (1993).
9) Poole, D.: A Logical Framework for Default Reasoning, *Artificial Intelligence*, Vol.36, pp.27–47 (1988).
10) Poole, D.: Probabilistic Horn abduction and Bayesian networks, *Artificial Intelligence*, Vol.64, pp.81–129 (1993).
11) 井上克巳：アブダクションの原理，人工知能学会誌，Vol.7, No.1, pp.48–59 (1992).
12) 伊藤史朗，石塚　満：推論パスネットワークによる高速仮説推論システム，人工知能学会誌，Vol.6, No.4, pp.501–509 (1991).
13) 加藤昇平，世木博久，伊藤英則：プログラム解析に基づく仮説推論の高速化技法，情報処理学会論文誌，Vol.35, No.10, pp.2019–2028 (1994).
14) 加藤昇平，世木博久，伊藤英則：コストに基づく仮説推論における最適解探索の一方法，情報処理学会論文誌，Vol.36, No.10, pp.2380–2390 (1995).
15) 近藤朗子，石塚　満：述語論理知識を扱う仮説推論における最適解の高速推論法，人工知能学会誌，Vol.9, No.2, pp.110–118 (1994).

(平成 11 年 3 月 25 日受付)
(平成 11 年 12 月 2 日採録)

加藤　昇平（正会員）

1993 年名古屋工業大学工学部電気情報工学科卒業．1995 年同大学院博士前期課程修了．1998 年同大学院博士後期課程修了．同年豊田工業高等専門学校助手，現在同講師．工学博士．高次推論，論理プログラム，画像処理等に興味を持つ．人工知能学会会員．

世木　博久（正会員）

1979 年東京大学工学部計数工学科卒業．1981 年同大学院工学系研究科修士課程修了．同年 4 月より三菱電気（株）中央研究所に勤務．1985 年〜1989 年（財）新世代コンピュータ技術開発機構に出向．1992 年 4 月より名古屋工業大学知能情報システム学科．現在同学科教授．工学博士．論理プログラミング，演繹データベース等に興味を持つ．電子情報通信学会，人工知能学会，ACM，IEEE Computer Society 各会員．

伊藤　英則（正会員）

1974 年名古屋大学大学院工学研究科博士課程電気・電子専攻満了．工学博士号取得．同年日本電信電話公社入社，横須賀研究所勤務．1985 年（財）新世代コンピュータ技術開発機構出向．1989 年より名古屋工業大学教授，現在知能情報システム学科所属．これまでに，数理言語理論とオートマトン，計算機ネットワーク通信 OS，知識ベースシステム等の研究と開発に従事．電子情報通信学会，人工知能学会，ファジー学会各会員．