論　文

ワークステーションクラスタを用いた並列仮説推論システム

加藤　昇平[†]　　中村　智典[††*]　　伊藤　英則[††]

A Parallel Cost-Based Abductive Reasoning System on Workstation Cluster

Shohei KATO[†], Tomonori NAKAMURA[††*], and Hidenori ITOH[††]

あらまし　本論文では，ワークステーションクラスタ環境で効率良く動作する動的負荷分散の一手法を提案し，同手法を用いた並列仮説推論システムを提案する．ワークステーションクラスタ環境では，処理能力や負荷の異なるワークステーションを用いて並列処理を行うため，それぞれのワークステーションの性能や負荷状態を考慮に入れた動的な負荷分散が要求される．本論文では，動作中のワークステーションの処理能力を推定するための方法と，推定された状態を均一にするための動的負荷分散アルゴリズムを提案する．また，同手法を用いて，与えられた観測に対する最も好ましい説明を効率的に求める並列仮説推論システムを実装し，その実験結果についても報告する．

キーワード　コストに基づく仮説推論，WS クラスタ並列，動的負荷分散

1. まえがき

ワークステーションクラスタ型並列処理（以降，WS クラスタ並列処理）とは，ネットワーク接続された複数のワークステーションで構成された分散システム上で行う並列処理である．WS クラスタ並列処理は，既存の資源（ワークステーション，PC，イーサネット等）を用いて手軽に並列処理が可能なため，近年注目を集めている技術である [1]．しかしながら同処理においては，並列計算機を用いた既存の並列処理とは異なる特徴が多く，その特徴を考慮したアルゴリズムが不可欠である．例えば，既存の並列計算機においては，プロセッサの性能はすべて等しく，1 人のユーザがプロセッサを占有できることが一般的である一方，WS クラスタ環境では，どのようなワークステーションを用いるかをユーザが自由に選択できるため，プロセッサのアーキテクチャや処理能力等に差異が生じる場合がある．また，WS クラスタ環境を構成する個々のワークステーションは，通常，複数のユーザによって自由に使用されるため，ワークステーションにかかる負荷も動的に変化する．このため，WS クラスタ並列処理では，各計算機がもつ計算能力や通信性能などの特性や動的に変化する負荷や通信コストなどに応じて，それぞれのワークステーションに負荷をどのように分散するかが極めて重要である．

仮説推論 [2], [3] は，不完全な知識のもとで適切な推論を行う高次推論の一形式であり，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って推論を進めるなど，柔軟で知的な知識処理を行うことができる．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極めて大きいことが知られており [4], [5]，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（例えば，文献 [6]〜[9]）．

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．現在までに，文献 [10]〜[12] において，仮説推論の探索空間を複数のプロセッサに分割し並列に推論を行う並列仮説推論システムを提案してきた．同システムは，並列ヒューリスティック探索がも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．しかし，このシステムは特定の並列計算機上でしか動作しないという問題点がある．

[†] 豊田工業高等専門学校電気・電子システム工学科，豊田市
Department of Electrical and Electronic Engineering, Toyota National College of Technology, Toyota-shi, 471–8525 Japan
[††] 名古屋工業大学知能情報システム学科，名古屋市
Department of Intelligent and Computer Science, Nagoya Institute of Technology, Nagoya-shi, 466–8555 Japan
* 現在，松下電器産業株式会社

そこで本論文では，ワークステーションクラスタ環境で有効に機能する動的負荷分散の一手法を提案することにより，高速な仮説推論処理システムを実現する．

2. コストに基づく仮説推論

本研究では，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 [14], [15] 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．また，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 1］ $\mathcal{F}$ をホーン節集合（事実），$\mathcal{H}$ を単位節の集合（仮説集合）とする．また，$\mathcal{O}$ を存在束縛されたアトムの連言（観測）とする．このとき，$\mathcal{O}$の $\mathcal{F}\cup\mathcal{H}$ による最適な説明$\mathcal{E}$とは，以下の条件を満足するような$\mathcal{H}$の要素の代入例からなる集合である．

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

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

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

$cost(\mathcal{H})$ は仮説集合$\mathcal{H}$(注1)に含まれる仮説のコストの和を表す．　□

ここで，$\mathcal{F}$ は常に成り立つ知識であるのに対し，$\mathcal{H}$ の代入例からなる部分集合は $\mathcal{F}$ と矛盾する可能性がある．また，$\mathcal{F}$と$\mathcal{H}$の和集合をプログラムと呼び，$\mathcal{P}$で表記する．

［定義 2］ $\mathcal{F}$の中にある $\mathit{false}\leftarrow A_1,\cdots,A_n$ で表現される節を「制約節」と呼ぶ．ただし，$A_k(1\le k\le n; n\ge 1)$ はアトムであり，false は矛盾を表す．　□

与えられたプログラム及び観測に対して無矛盾な説明を求める処理では，観測の負節に対する SLD 反駁(ばく) [16] に簡単な変更を加えることにより [6]，これを実現している．詳しくは文献 [6] を参照されたい．

本研究では，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，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している．

［定義 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．

$$f(g)=cost(\{L_1,\cdots,L_i\})+\hat{h}(g) \quad (1)$$

$\hat{h}(g)$ は以下の条件を満足するような予測値とする．

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

$h(g)=cost(\bigcup_{j=i+1}^{n}\mathcal{E}_{A_j})$

ただし，$L_k(1\le k\le i)$，$A_j(i+1\le j\le n)$ はアトムであり，$\mathcal{E}_{A_j}$ は A_j の最適な説明を表す．　□

ここで，ゴール節に現れる M なるオペレータが付いたアトム “ML_k” は，サブゴール $\leftarrow L$（ここで L_k は L 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，L 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．したがって，上記のゴール節 g において$\{L_1,\cdots,L_i\}$は反駁木の根から g に至る導出過程で立てられた仮説集合を表しており，$cost(\{L_1,\cdots,L_i\})$ は同集合のコストの値を示す．また，本研究では，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．そこで本研究では，SLD 反駁に対して A* アルゴリズムがもつ探索制御技術を導入することにより [9]，効率的な仮説推論システムを実現している．推論方法についての詳細は文献 [9] を参照されたい．

3. 並列仮説推論システム

先行研究 [10]〜[12] において，前章で述べた推論及び探索制御技術の並列化を提案し，コストに基づく並列仮説推論システムを実現した．しかしながら同システムは特定の並列計算機上でしか動作しないという問題点があった．そこで本論文では，WS クラスタ環境上で効率的に動作する並列仮説推論システムを提案する．仮説推論の探索空間を仮想並列計算機上の複数のプロセッサへ分散し，各プロセッサが同期，通信を繰り返し並列に推論することにより，最適な説明，すなわちコスト最小の説明（以下，最適解と呼ぶ）を効率的に求めるものである．本章では，まず，システムの構成を概説し，仮想並列計算機がもつプロセッサ性能・負荷状態の不均衡を解消するための動的な負荷分散手法を提案し，同手法を組み込んだ並列仮説推論システムのアルゴリズムを説明する．

(注1)：$\mathcal{H}=\phi$ のときは $cost(\mathcal{H})=0$ とする．

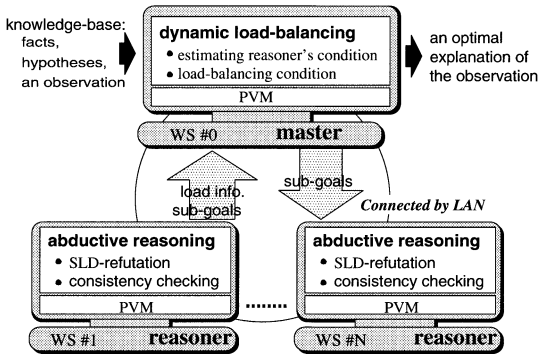

図 1　並列仮説推論システムの概略
Fig. 1　An overview of our parallel cost-based abductive reasoning system.

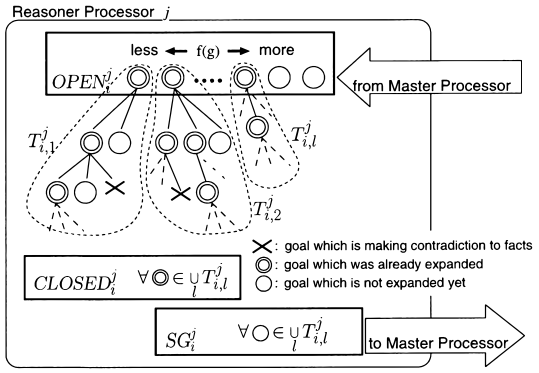

図 2　reasoner における局所推論の模式図（反復 i 時）
Fig. 2　A model of local reasoning in a slave processor at i-th iteration.

3.1　システム構成

図 1 に本並列仮説推論システムの概略を示す．本システムは，1 台の master 及び N 台の reasoner で構成され，PVM (Parallel Virtual Machine)[13] を用いて構築された仮想並列計算機上で動作する．PVM は，ネットワークで接続された異機種のワークステーション群を単一の仮想的な分散メモリ型並列計算機とみなし，並列処理プログラムの実装を可能にするプロセッサ間通信ライブラリである．

本システムにおいて，各 reasoner は，仮説推論を局所的に行い，局所推論により得られたサブゴール節（形成された推論木(注2)の葉にあたるゴール節）と自分の負荷情報を master へ送信する．master は，まず，全 reasoner からサブゴール節と負荷情報を収集し，得られた情報から各 reasoner の状態を推定する．そして，各 reasoner の状態に基づいてサブゴール節を全 reasoner に分散する．この処理の反復により，コストに基づく並列仮説推論が実行される

3.2　reasoner の状態に基づく動的負荷分散

3.2.1　reasoner の状態の推定

仮想並列計算機には，処理速度や負荷状態の異なる reasoner が混在する．そのため，効率良く仮説推論を実行するためには，処理速度の速い reasoner や負荷の軽い reasoner により多くの探索空間を割り当てるべきである．

reasoner の処理速度は，CPU の性能やメモリの搭載量から静的に測定可能であるが，reasoner にかかる負荷については，それが他のユーザが発生したプロセスなどによってシステムとは無関係に生じることや，時間とともに動的に変化する点から，あらかじめ測定することは不可能である．

本並列仮説推論システムでは，全 reasoner の局所推論は同期がとられており，同期の間隔は一定時間に固定されている．各 reasoner が処理する局所的な推論では，同間隔の間最良優先の SLD 反駁が実行される．図 2 に同推論の模式図を示す．同推論では，展開キュー（図 2 中 $OPEN_i^j$ に相当）及び各 SLD 反駁木の葉にあたるサブゴール節（図 2 中○で示されたノードに相当）に対して，評価値が良いものから優先的に 1 ステップずつ SLD 導出が行われるため，特定の reasoner に評価値の良いサブゴール節が集中することは避けるべきである．

そこで，本研究では「reasoner の処理能力」「reasoner におけるゴール節のふくそう」の 2 点に着目することにより，reasoner の状態を定義し，動的な負荷分散の評価値に用いている．

［reasoner の処理能力 (ability)］

ある時点での reasoner の処理能力は，前回の負荷分散によりサブゴール節を受信してからその時点までに（すなわち 1 回の反復において）reasoner が行った仕事の量に基づき推定される．仕事の量は 1 回の反復における局所推論(注3)で reasoner が生成した部分 SLD 反駁木の大きさとする．

［定義 4］　i 回目の反復における reasoner j の処理能力の推定値は以下の式で定義される．

(注2)：局所推論において形成される反駁木の深さ（推論ステップ）は固定されておらず，局所推論の終了は master から発信される同期信号によって制御される．
(注3)：本並列仮説推論システムにおいては，全 reasoner の局所推論は同期がとられており，同期の間隔は一定時間に固定されている．

$$ability_i^j = \frac{\sum_l |T_{i,l}^j|}{\sum_{s=1}^{N} \sum_l |T_{i,l}^s|} \quad (2)$$

ここで，$T_{i,l}^j$ は i 回目の反復で reasoner j が展開した部分 SLD 反駁木（l は識別子）であり，$|T_{i,l}^j|$ は $T_{i,l}^j$ に含まれるゴール節の数である．□

reasoner が行う局所推論では，図 2 に示すように複数の SLD 反駁木が生成される点に注意されたい．

［ゴール節のふくそう (convergence)］

ある reasoner に評価値の良いゴール節がどの程度集中しているかの度合は，サブゴール節を再分配するために master が reasoner から受け取ったゴール節を用いて以下のように推定される．

［定義 5］ CS_i^j を i 回目の反復時に master が reasoner j から受信したサブゴール節（図 2 中集合 SG_i^j に相当）の中で最良 $\alpha \times ability_i^j$ 個（小数切上げ）のサブゴール節(注4)からなる集合（convergence set と呼ぶ）とする（α はパラメータ）．このとき，i 回目の反復時における reasoner j に対するゴール節のふくそう $convergence_i^j$ は以下の式で定義される．

$$convergence_i^j = \frac{\sum_{g \in CS_i^j} f(g)}{|CS_i^j|} \quad (3)$$

ここで，$f(g)$ はゴール節 g の評価関数（定義 3 参照）であり，$|CS_i^j|$ は CS_i^j の要素の数である．□

［reasoner の状態］(condition)

reasoner の状態は上記の評価値を用いて次のように定義される．

［定義 6］ i 回目の反復における reasoner j の状態の推定値は以下の式で定義される．

$$condition_i^j = ability_i^j \times convergence_i^j \quad (4)$$

□

master が実行する負荷分散処理では，上記により求められた reasoner の *condition* がすべての reasoner でなるべく均一になるようにスケジューリングされ，各 reasoner から収集されたサブゴール節が再分配される．

3.2.2 reasoner の状態に基づく動的負荷分散

図 3 に *condition* を用いた動的な負荷分散のアルゴリズムを示す．同図において，$Reasoners, SG_i^j$，CS_i^j はそれぞれ，reasoner の ID の集合，i 回目の反復において reasoner j から受信したサブゴールの集合，同反復における reasoner j の *convergence set* を表す．

distribute_goals(SG, CS, t)
入力: 全 reasoner から受信したサブゴール節の集合 SG,
reasoner の *convergence set* の集合 CS,
現時点での最適解候補がもつコスト t.

```
1  begin
2    for each j ∈ Reasoners
       condition_i^j 及び ability_i^j をそれぞれ算出;
3    while (Reasoners ≠ φ and SG ≠ φ)
4      begin            /* condition に基づく負荷分散 */
5        G_min := ⋃_s SG_i^s の中で評価値最小のゴール節;
6        k := ∃s ∈ Reasoners | G_min ∈ SG_i^s;
7        j := ∃s ∈ Reasoners | condition_i^s の値が最大;
8        G_min を reasoner j へ分散する;
         /* 分配先の reasoner の状態を更新 */
9        condition' := condition_i^j;
10       CS_i^j ∪ {G_min} に含まれる最良 |CS_i^j| 個のゴール
         節で CS_i^j を更新する;
11       CS_i^j の内容から convergence_i^j を更新する;
12       condition_i^j := ability_i^j × convergence_i^j;
13       if (j ≠ k and condition_i^j = condition') then
          begin
14         G := CS_i^j ∩ SG_i^j;
15         G に含まれるゴール節を reasoner j へ分散する;
16         SG_i^j := SG_i^j \ G;
17         Reasoners := Reasoners \ {j};
18        end
19       /* 分配元の reasoner の状態を更新 */
         SG_i^k := SG_i^k \ {G_min};
20       SG_i^k に含まれる最良 |CS_i^k| 個のゴール節で CS_i^k を
         更新する;
21       CS_i^k の内容から convergence_i^k を更新する;
22       condition_i^k := ability_i^k × convergence_i^k;
23     end
24   if (⋃_s SG_i^s ≠ φ) then /* ability に基づく負荷分散 */
       distribute_goal_by_ability(⋃_s SG_i^s) ;
25 end.
```

図 3 *condition* による負荷分散手法
Fig. 3 Dynamic load-balancing by *condition*.

本負荷分散アルゴリズムでは，*condition* の値が最大となる 1 台の reasoner に評価値が最小な 1 個のサブゴールを分配することが繰り返される．reasoner k から master が受信したサブゴール g が reasoner j へ分散されるとする．このとき，k 及び j の *convergence set* は次のように変化する．

$$CS_i^k := SG_i^k \setminus \{g\} \text{ に含まれる最良 } |CS_i^k| \text{ 個のゴール節} \quad (5)$$

$$CS_i^j := CS_i^j \cup \{g\} \text{ に含まれる最良 } |CS_i^j| \text{ 個のゴール節} \quad (6)$$

i 回目の反復における本負荷分散の初期段階では，各

(注4)：reasoner j が送信したサブゴールの数が $\alpha \times ability_i^j$ よりも少ない場合は，j が送信したすべてのサブゴールの集合を CS_i^j とする．

reasoner に対する *convergence set* は i 回目の局所推論において導出された優良なゴール節を保持しており，*convergence* の値はゴール節の輻輳の程度を示している．同値と *ability* の値を reasoner の状態の初期値に与えてゴール節分散を開始することにより，本負荷分散は，他の reasoner で導出された優良なゴール節を，i 回目の局所推論において優良なゴール節が不足した，あるいは，処理能力が高かった reasoner へ優先的に分配する．そして，ゴール節の分配が繰り返されることにより，*convergence set* の内容は $i+1$ 回目の局所推論のために分散されるサブゴール節へ変化していく．

本アルゴリズムでは，定義 5 より CS_i^j の要素数は $\alpha \times ability_i^j$ に固定されており，また，評価値の良いゴール節から分配先が決定される．したがってゴール節分散を繰り返すにつれて，reasoner j に新たにゴール節を分散しても CS_i^j 及び $condition_i^j$ が変化しない状態に落ち着く．このような状態(注5)になると reasoner j を $Reasoners$ から削除する．$Reasoners$ が空集合になると reasoner 間の状態の不均衡が解消されたと考え，*condition* に基づく負荷分散は終了する．本負荷分散の終了時には，各 reasoner の *convergence set* は $i+1$ 回目の局所推論において展開されるゴール節を保持するようになる．このとき，まだ SG にサブゴール節が残っている場合は，それらを $ability_i^j$ に応じて分散する．

distribute_goals_by_ability(SG)
入力: 全 reasoner から受信したサブゴール節の集合 SG.

```
 1 begin
 2   for each j ∈ Reasoners
       total_i^j := 0;
 3   repeat
 4     for each s ∈ Reasoners
 5       begin
 6         total_i^s := total_i^s + ability_i^s;
 7         if (total_i^s ≥ 1) then
 8           SG の中で評価値が最良なゴール節を
             G_min とする;
 9           G_min を reasoner s へ分散する;
10           SG := SG \ {G_min};
11           total_i^s := total_i^s − 1;
12         end
13       end
14   until (SG = φ)
15 end.
```

図 4　*ability* による負荷分散手法
Fig. 4　Dynamic load-balancing by *ability*.

図 4 に $ability_i^j$ に基づく負荷分散アルゴリズムを示す．本アルゴリズムでは，各々の reasoner に対して $ability_i^j$ の大きさに比例した量のゴール節が，評価値の良いものから順に分散される．このアルゴリズムはすべてのゴール節を分散するまで終了しない．これにより master が収集した全サブゴール節はいずれかの reasoner に分配される．

3.3　master のアルゴリズム

master のアルゴリズムを図 5 に示す．同図において，SG_i，$|OPEN_i^j|$，ANS はそれぞれ i 回目の反復ですべての reasoner から受信したサブゴールの集合，reasoner j 内でまだ展開されていないゴール節の数，発見された最適解の候補を表すゴール節の集合を示している．また，t は現時点までに得られた適解の候補のコストを表し，N は reasoner の台数を表す．

master は，reasoner を起動し「サブゴール節分散手続き」及び「サブゴール節収集手続き」の二つの手続きを繰り返す．

サブゴール節分散手続きでは，**3.2** で説明した動的負荷分散手法により SG_i を reasoner へ分散する．こ

Parallel Cost-based Abductive Reasoning (master)
入力:プログラム $\mathcal{P}$, 観測 $\mathcal{O}$.
出力:最適な説明（成功裏）または false (失敗裏).

```
 1 begin
 2   SG_0 := {← O}, i := 0, t := ∞;
 3   Reasoners := {reasoner^1, ··· , reasoner^N};
 4   spawn_reasoners(Reasoners, P);
 5   repeat
 6     /* サブゴール節分散手続き */
       distribute_goals(SG_i, CS_i, t);
 7     i := i + 1;
 8     /* サブゴール節収集手続き*/
       from each reasoner j
         begin
10         collect_goals(SG_i^j, |T_i^j|, |OPEN_i^j|, ANS, t);
11         CS_i^j := { 最良α個の g | g ∈ SG_i^j };
12         CS_i := CS_i ∪ {CS_i^j};
13         SG_i := SG_i ∪ {SG_i^j};
14       end
15   until (Σ_{j=1}^{N} |OPEN_i^j| = 0 and SG_i = φ)
16   kill_reasoners(Reasoners);
17   if (ANS ≠ φ) then return ANS;
18   else return false;
19 end.
```

図 5　master のアルゴリズム
Fig. 5　Algorithm for a master processor.

(注5)：CS_i^j に含まれる（すなわち reasoner j で導出された）ゴール節 g が再び reasoner j へ分配される場合，CS_i^j 及び $condition_i^j$ は変化しないが，このような場合は例外とする．

のとき，t の値も全 reasoner へ送信する．

サブゴール節収集手続きでは，すべての reasoner からの通信を待ち，サブゴール節，及び，サブゴール節分散手続きでのスケジューリングに必要な情報をそれぞれ受信する．reasoner から最適解の候補が送られてきた場合には，それを ANS へ加え，t にそのコストを代入する．複数の reasoner から同時に最適解の候補が送られてきた場合には，それらうちで評価値が最小のものを ANS へ加え，そのコストを t に代入する．

すべての reasoner において未展開ゴール節がなくなると，master 内の手続きの反復を終了し，reasoner に推論を終了するように命令を送信する．そして，ANS に含まれている解のうち，コストが最小のものを最適解として出力し，成功裏に終了する．このとき，$ANS = \phi$ の場合には推論は失敗裏に終了する．

3.4 reasoner のアルゴリズム

図 6 に，reasoner のアルゴリズムを示す．同図において，SG，$OPEN$，$CLOSED$ はゴール節の集合を表す（各集合の役割は図 2 参照）．また，ANS は reasoner において導出された最適解の候補を表すゴール節の集合を表す．reasoner のアルゴリズムは，「ゴール節展開手続き」，「サブゴール節受信手続き」，及び，「サブゴール節送信手続き」の三つの手続きの反復からなる．ある reasoner j での i 回目の反復は以下のように動作する．

Parallel Cost-based Abductive Reasoning (reasoner)
入力: プログラム$\mathcal{P}$.

```
 1 begin
 2   OPEN := φ, SG := φ, ANS := φ, t = ∞;
 3   work = 0, exit = false;
 4   repeat
 5     /* ゴール節展開手続き */
       SG := φ, ANS := φ, CLOSED := φ;
 6     reset_timer(); timeout := false;
 7     repeat
 8       Gmin := {∀g ∈ OPEN ∪ SG |
                ∀g' ∈ OPEN ∪ SG : f(g) ≤ f(g')};
 9       RS := Gmin 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の集合;
10       RS 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 から削除する;
11       評価値が t より大きなゴールを RS から削除する;
12       ANS := {∀g ∈ RS |
                  g は f(g) が最小の成功葉 };
13       SG := SG ∪ RS;
14       CLOSED := CLOSED ∪ Gmin;
15       OPEN := OPEN \ Gmin;
16       SG := SG \ Gmin;
17     until (timeout)
18     work := |SG| + |CLOSED|;
19     /* サブゴール節送信手続き */
       send_goals_to_master(ANS, SG, work, |OPEN|);
20     /* サブゴール節受信手続き */
       receive_goals(OPEN', t, exit);
21     OPEN := OPEN ∪ OPEN';
22     OPEN から評価値が t より大きなゴールを削除する;
23   until (exit)
24 end.
```

図 6　reasoner のアルゴリズム
Fig. 6　Algorithm for reasoner processors.

ゴール節展開手続きでは，一定の時間（図 6 のアルゴリズム中 $timeout$ により設定）仮説推論が実行され，$OPEN$ に含まれるゴール節を根とするような部分 SLD 反駁木がゴール節の評価値に対する最良優先で生成される．得られたサブゴールに対する無矛盾性の検査は，あらかじめに制約節に対する SLD 反駁木を求めておき，その成功葉に現れる仮説集合がサブゴール中に含まれるかどうかで判定する(注6)．

サブゴール節送信手続きでは，展開の結果得られたサブゴール節集合 SG と最適解の候補 ANS を master へ送信する．このとき，同時にゴール節展開手続きで生成した SLD 反駁木の大きさと，$OPEN$ に含まれるゴール節の数を送信する．

ゴール節の受信手続きでは，master から受け取ったゴール節を $OPEN$ へ追加し，t の値を更新する．このとき，$OPEN$ に含まれるゴール節 g のうち $f(g) > t$ となる g を $OPEN$ から削除する．また，master から推論終了の命令が届いた場合には $exit$ が $true$ にセットされ，reasoner のアルゴリズムは終了する．

4. 実　　験

本論文で提案した手法の有効性を確認するために，**3.** で提案した並列仮説推論システムに，**3.2** で提案した動的負荷分散手法を実装して推論を実行し，推論の実行時間，及び，並列効果について調べた．実験は SUN Ultra-1 をイーサネットで接続して構築した WS クラスタ計算機上で行った．また，同クラスタにおいて 1/4 の reasoner にはある一定の負荷を与え，reasoner 間の処理性能に差を付けた．

仮説推論の例題には，n-bit 全加算器故障診断問題を用いた．回路を構成する論理素子は「正常」「開放」「短絡」の三つの状態をとるものとし，これを仮説として表現した．システムの実装には，master においては C

(注6)：サブゴールには，それまでの推論過程で生成された仮説がすべて保持されるため（定義 3 参照），reasoner ごとに独立した無矛盾性の検査が可能である．

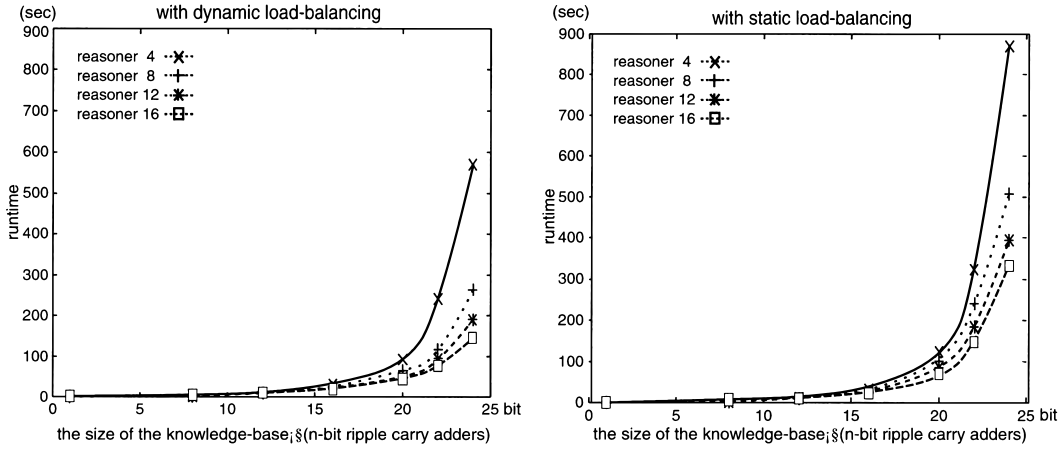

図 7 実験結果
Fig. 7 Experimental results.

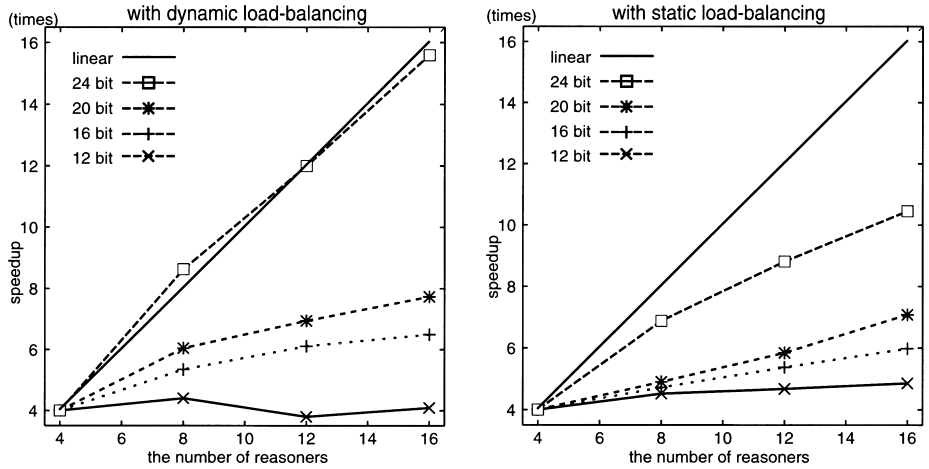

図 8 速度向上比
Fig. 8 Speedups.

言語，reasoner においては並列論理型言語 KLIC [17] をそれぞれ用いた．master がゴール節を分散してから同期信号を送信するまでの間隔（各 reasoner が行う一回の推論時間に相当）は 2300 (ms) とした．動的負荷分散のパラメータ（定義 5 参照）は $\alpha = 5 \times N$（N は reasoner 台数）とした．また，探索木に現れるゴール節 g のコストの予測値（定義 3 参照）は $\hat{h}(g) = 0$ とした．以上の条件で，問題規模 n(注7)，reasoner 数 N を変化させて推論時間の比較実験を行った．推論システムの比較対象としては，静的負荷分散手法(注8)が組み込まれた並列仮説推論システムを用いた．

図 7 はその結果である．動的負荷分散を適用したシステムの方が推論時間が短縮されていることがわかる．図 8 にそれぞれの速度向上比を示す．ここでは，それぞれの reasoner 数における推論時間を reasoner 数 4 の推論時間で割った値を 4 倍した(注9)．なお，推論

(注7)：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 $17n + 23$ のホーン節，仮説は $15n + 3$ の単位節からなる．
(注8)：動的負荷分散に用いる変数 $condition$（定義 4 参照）を定数にすることで実現できる．
(注9)：本実験では 1/4 の reasoner に過負荷を与えているため，reasoner を 4 台未満にした状態で同一の実験環境が実現できない．また，4 台未満の WS クラスタ環境ではシステム全体の計算能力が不足し規模 n の大きな問題が解けなかった．

表 1　$n = 24$ のときの実験結果の詳細
Table 1　The detail of the results on the problem with $n = 24$.

	with dynamic load-balancing			
reasoners	4	8	12	16
runtime (sec)	570.0	264.0	190.1	146.7
$\|CLOSED\|$	61572	69652	83380	91720
	with static load-balancing			
reasoners	4	8	12	16
runtime (sec)	870.8	507.5	395.2	333.5
$\|CLOSED\|$	82676	106244	112760	131831

表 2　reasoner 能力の不均衡による影響
Table 2　Influence of imbalance ability of slaves to the system.

	with dynamic load-balancing		
reasoners	15	$16'$	16
runtime (sec)	118.5	112.3	108.0
$\|CLOSED\|$	83480	85720	79988
	with static load-balancing		
reasoners	15	$16'$	16
runtime (sec)	124.6	184.7	118.2
$\|CLOSED\|$	84820	97180	83052

時間にはプロセッサ間の通信時間や負荷分散の計算時間も含まれている．同図より，静的負荷分散を用いたシステムでは，reasoner 数が増加するにつれて台数効果は線形から外れていく一方，動的負荷分散を適用した本システムでは，知識ベースの規模が大きな場合には，reasoner 台数が増加してもほぼ線形の速度向上が実現されていることがわかる．$n = 24$ における実験結果の詳細を表 1 に示す．同表において $|CLOSED|$ はシステムが最適解を求めるまでに展開したゴール節の数を示しており，探索空間の大きさに相当する．$|CLOSED|$ より，動的負荷分散を用いた方が探索空間が小さく抑えられており，本負荷分散手法により無駄な探索が削減されていることがわかる．

次に，WS クラスタ内の一部の reasoner に負荷がかかった場合のシステム全体への影響を調べた．表 2 に $n = 24$ の問題を N 台，$N-1$ 台，N' 台の reasoner で解いた実験結果を示す．ここで N' 台は N 台の reasoner 中 1 台 が過負荷状態にあることを表す．

システム全体の処理能力をすべての reasoner の処理能力の総和であると考えると，システムの実行時間は以下の不等式(注10)を満たすと考えられる．

$$time(N-1) \geq time(N') > time(N) \qquad (5)$$

表 2 から，本論文で提案した動的負荷分散を用いた場合には本並列仮説システムは式 (5) を満足していることがわかる．したがって，一部の reasoner に負荷が与えられることによるシステム全体のパフォーマンスの低下は少ないといえる．一方，静的負荷分散を用いた実験結果では，$16'$ 台の推論において，15 台よりも台数が増加したにもかかわらず，長い推論時間を費やしていることがわかり，式 (5) は満足されない．静的負荷分散では，一部の reasoner に負荷を与えることによりシステム全体のパフォーマンスが大きく低下してしまうといえる．

また，負荷を与えたときの探索空間の大きさ $|CLOSED|$ を調べると，静的負荷分散を用いたシステムでは，探索空間が大きく増加しているのに対し，動的負荷分散を用いたシステムでは探索空間の増大を抑えていることがわかる．

以上より，本実験においては，本論文で提案した動的負荷分散を適用することで，過負荷状態の reasoner が混在している WS クラスタ環境においても，システム全体のパフォーマンスを急激に落とすことなく推論が実行された．

5. むすび

本論文では，WS クラスタ環境で動作する並列仮説推論システムを提案し，同推論システムにおいて，WS クラスタ環境に効果的な負荷分散の一手法を提案した．本手法は，推論実行中の各プロセッサの状態を推定し，推定値に基づき負荷を動的に分散する方法である．本負荷分散を搭載した並列仮説推論システムを実装し，実験によりその有効性を確認した．WS クラスタ環境における負荷分散の阻害要因としては，**3.2.1** で述べたように，「プロセッサ内のゴール節の輻輳 (convergence)」及び「プロセッサ 間の処理能力 (ability) の差異」に起因するプロセッサの状態 (condition) の不均衡が考えられる．本論文で提案した負荷分散方法は，まず，*condition* を用いた動的な負荷分散によって前者の阻害要因を緩和し，続いて，*ability* に基づく負荷分散によって後者の阻害要因に対処するものである．本論文で提案した負荷分散のアルゴリズムはプロセッサ間の状態の厳密な均一化を保証するものではない．1 回の局所推論でプロセッサが導出した全ゴール節を考慮に入れた *condition* 値を考え，整数計画法の近似アルゴリズムを用いればプロセッサの状態を均一化する負荷分散は可能だが，負荷

(注10)：同式において，$time(N)$ は N 台の reasoner で構成されたシステムの推論時間を示す．

分散の計算コストや通信コストが増大し，本並列仮説推論システムでは，システム全体のパフォーマンスに対する影響が大きいと考えた．そこで本研究では，なるべく簡単な分散スケジューリングでプロセッサ間の状態の不均衡を抑える方法を提案した．今後の課題として，プロセッサ間の状態の均一化を保証しつつ，計算コストや通信コストの負担が少ない負荷分散アルゴリズムを考案したい．

本論文では，WS クラスタ環境における仮説推論の並列化において，負荷の分散方法を中心に本並列仮説推論システムを説明した．本論文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，$\hat{h}(g)$ の事前解析方法及び $\hat{h}(g)$ の精度向上による推論制御の効果については文献 [9] を参照されたい．

謝辞 本研究は，一部，(財) 内藤科学技術振興財団，及び，科学研究費奨励研究 (A) (課題番号 12780297) の助成により行われた．

文　献

[1] 森眞一郎，富田眞治，“並列計算機アーキテクトからみた計算機クラスタ，” 情報処理，vol.39, no.11, pp.1073–1077, 1998.
[2] D. Poole, R. Goebel, and R. Aleliunas, “Theorist: A logical reasoning system for defaults and diagnosis,” in The Knowledge Frontier: Essays in the Representation of Knowledge, ed. N. Cercone and G. McCalla, pp.331–352, Springer-Verlag, 1987.
[3] 井上克巳，“アブダクションの原理，” 人工知能誌，vol.7, no.1, pp.48–59, 1992.
[4] B. Selman and H.J. Levesque, “Abductive and default reasoning: A computational core,” Proc. AAAI-90, pp.343–348, 1990.
[5] 石塚　満，“仮説推論の計算量と高速化メカニズム，” 人工知能誌，vol.9, no.3, pp.342–349, 1994.
[6] 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，” 情処学論，vol.35, no.10, pp.2019–2028, 1994.
[7] 石塚　満，“仮説推論の計算量と高速化メカニズム，” 人工知能誌，vol.9, no.3, pp.342–349, 1994.
[8] 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，” 人工知能誌，vol.9, no.2, pp.110–118, 1994.
[9] 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，” 情処学論，vol.36, no.10, pp.2380–2390, 1995.
[10] S. Kato, H. Seki, and H. Itoh, “Parallel cost-based abductive reasoning for distributed memory systems,” Lecture Notes in Artif. Intell., vol.1114, pp.300–311, Springer-Verlag, Cairns, 1996.
[11] S. Kato, H. Seki, and H. Itoh, “A parallel implementation of cost-based abductive reasoning,” Proc. PASCO’97, pp.111–118, ACM Press, 1997.
[12] 加藤昇平，世木博久，伊藤英則，“PARCAR: コストに基づく並列仮説推論システム，” 情処学論，vol.41, no.3, pp.668–676, 2000.
[13] V. Sunderam, “Pvm: A framework for parallel distributed computing,” Concurrency: Practice and Experience, vol.2, no.4, pp.315–339, 1990.
[14] D. Poole, “Probabilistic Horn abduction and Bayesian networks,” Artif. Intell., vol.64, pp.81–129, 1993.
[15] E. Charniak and S.E. Shimony, “Cost-based abduction and MAP explanation,” Artif. Intell., vol.66, pp.345–374, 1994.
[16] J.W. Lloyd, Foundations of Logic Programming, Springer, 1984. Second, extended edition, 1987.
[17] K.T. Fujise, T. Chikayama, and A. Nakase, “Klic: A portable implementation of kl1,” Proc. FGCS’94, 1994.

（平成 12 年 8 月 14 日受付，13 年 1 月 5 日再受付）

加藤　昇平　（正員）

1993 名工大・電気情報卒．1995 同大大学院工学研究科博士前期課程電気情報工学専攻了．1998 同大学院博士後期課程了．同年豊田工業高等専門学校助手，1999 同講師．工博．高次推論，論理プログラム，並列探索，画像処理等に興味をもつ．情報処理学会，人工知能学会各会員．

中村　智典

1997 名工大・知能情報システム卒．1999 同大大学院電気情報工学専攻博士前期課程修了．同年 4 月松下電器産業 (株) 入社．現在は，同社ソフトウェア開発本部にてホームネットワークの開発に従事する．在学中は，高次推論，並行論理プログラム，並列探索問題等に興味をもつ．

伊藤　英則　（正員）

1974 名大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．1974 日本電信電話公社横須賀研究所勤務．1985 (財) 新世代コンピュータ技術開発機構出向．1989 名古屋工業大学教授．現在知能情報システム学科所属．この間，数理言語理論，計算機ネットワーク通信，OS，知識ベースシステムなどの研究開発に従事．情報処理学会，人工知能学会，形の科学学会，日本ファジィ学会各会員．