並列推論実験ソフトウェア

新田克己　　瀧 和男　　市吉伸行

第7研究室
(財) 新世代コンピュータ技術開発機構
〒108 東京都港区三田 1-4-28
{nitta,taki,ichiyoshi}@icot.or.jp

概要

大規模な知的システムを開発するツールとしてICOTは並列推論マシン PIM、並列論理型言語 KL1、オペレーティングシステム PIMOS を開発した。これらの実用性を評価するため、われわれは KL1 を用いて4つの応用プログラム (LSI-CAD システム、遺伝子解析システム、法的推論システム、囲碁システム) を開発し、また関連のコンピュータメーカーに8つの応用プログラムの開発を委託した。これらの応用プログラムは事例ベース推論やモデルベース推論、定性推論、機械学習などの幅広い知識処理技術をカバーしている。

応用プログラムで高い計算効率を得るため、われわれは並行アルゴリズムや負荷分散などの並列プログラミング技術も開発した。さらに、その並列プログラミング技術の理論的解析を行った。これらの結果により、並列プログラミングの技術を選択するときの指針が得られた。

本論文では、それぞれの応用プログラムの概要と性能解析の結果を紹介し、並列プログラミング技術について検討している。

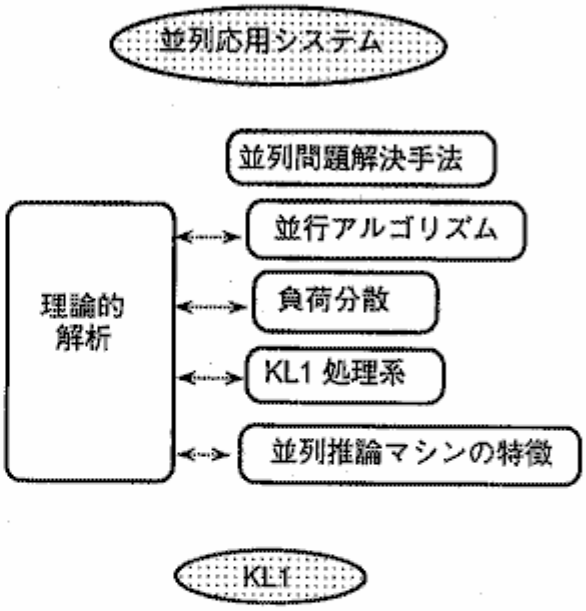


図 1: 並列プログラミングの技術

1　はじめに

知識処理システムを開発するツールとしてICOTは実験推論マシン Multi-PSI と5つのモデルの並列推論マシン PIM を開発した [Uchida *et al.* 1988] [Goto *et al.* 1988]。これらのマシンは MIMD 型のマシンであり、その上で並列論理型言語 KL1 で書かれたユーザープログラムが並列に実行される [Chikayama 1992]。KL1 は1階述語論理に基づいているので、人間の知識を自然に記述し推論過程を厳密に形式化することができる。従って、KL1 と PIM という組合せによって大規模な知的システムが比較的容易に開発できる。しかしながら、KL1 プログラムを不用意に書くと、期待した効率を得ることはできない。それは、効率が逐次的なボトルネックや並列化にともなうオーバーヘッドの影響を受けるからである。したがって、良い並列アルゴリズムや負荷分散の技術が開発されなければならない。また、効率良いプログラムを開発するためには、KL1 言語や PIM のアーキテクチャの性質を知っておかなくてはならない (図1)。これらの並列技術は密接に関連しているので、KL1 プログラムを開発するときに、どの技術を選択するかは慎重に行わなくてはならない。適切な技術を選択し、データサイズやプロセッサ数や効率との関係を予測するには指針が必要である。そのような指針を獲得するには、並列技術の理論的解析も必要となる。

われわれは以下の目標を達成するために並列の応用プログラムを開発した。

- PIM が実用的な知識システムの開発に適していることの実証:
  PIM は並列推論で高速に問題を解決するから、大規模なシステムの開発が比較的容易である。新しい応用

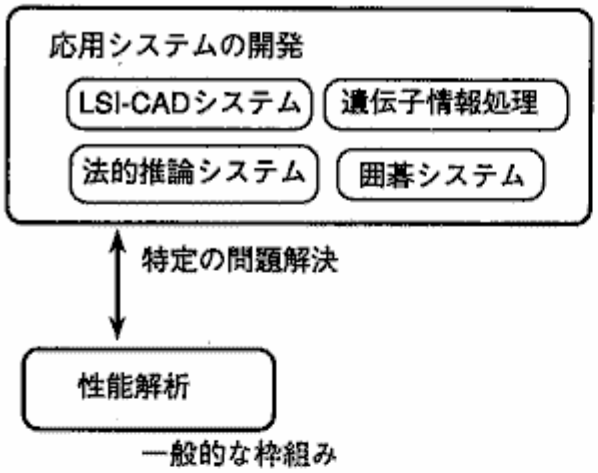


図 2: 第 7 研究室の研究体制

分野の開拓と、並列向きの知識処理技術を目指す。

- 並列プログラミング技術の開発：
  応用プログラムを解析することによって、高い計算効率が得られるための並列プログラミング技術を抽出できる。これらの技術の蓄積が新しい並列プログラムの開発を容易にする。

この論文では、第 2 節で ICOT 第 7 研究室の活動の概要を紹介する。第 3 節では第 7 研究室で開発された応用プログラムを紹介し、第 4 節では関連のコンピューターメーカーによる応用プログラムを紹介する。第 5 節では、性能解析の研究成果を紹介し、第 6 節では、並列プログラムの開発経験をまとめる。

2 第 7 研究室の活動

並列推論マシン PIM の上で応用プログラムを開発するには、専門家の知識からマシンの性質に至るまでの幅広い技術を必要とする。研究を効率良く進めるため、第 7 研究室の研究者を応用チームと性能解析チームに分けた（図 2）。応用チームは特定の応用プログラムを開発し、性能解析チームはプログラムを解析して並列プログラミングの指針を提供するのが目的である。

応用チームのテーマは次の 4 つである。専門家からの知識獲得を活発に行うためにこれらのチームは、並列 CAD(PIC)、遺伝子情報処理 (GIP)、高度設計 (ADS)、知識アーキテクチャ (KAR) の 4 つのワーキンググループを開催した。

- LSI-CAD システム
  LSI の設計は、アーキテクチャ設計や論理シミュレーションやレイアウト設計などのいくつかの段階からなる。このチームはそのうちで下流部分である論理シミュレーションとレイアウト設計を扱った。

- 遺伝子情報処理
  遺伝子解析の重要な目標の 1 つはタンパク質の配列の意味を解釈することである。このチームはタンパク質配列解析システム、タンパク質おれたたみシミュレーション、タンパク質構造解析プログラム、などを開発した。

- 法的推論システム
  法的推論の実現が困難なのは、用いられる用語があいまいだからである。あいまいな概念を扱うために、ルールベースと事例ベースを組み合わせたシステムを開発した。

- 囲碁システム
  囲碁は日本の伝統的なゲームである。このチームは並列版の囲碁システムを開発した。

この他に関連するコンピュータメーカーに以下の 8 つの応用システムの開発を委託した。

1. Co-HLEX: 協調再帰計算原理を用いた LSI レイアウト問題解決システム
   （階層的・協調型問題解決）
2. 協調型論理設計エキスパートシステム
   （仮説推論、協調型問題解決）
3. 事例に基づく回路設計支援システム
   （事例ベース推論）
4. 並列ルールベースアニーリングによるハイレベル合成
   （ルールベースアニーリング）
5. 深い推論に基づく設計支援システム
   （定性推論）
6. プラントモデルに基づく診断制御エキスパートシステム
   （モデルベース推論、定性推論）
7. 適応型モデルベース診断システム
   （モデルベース推論、機械学習）
8. モチーフ抽出システム
   （遺伝的アルゴリズム、機械学習）

これらは設計や診断や制御などの幅広い機能を実現し、事例ベース推論やモデルベース推論、定性推論や機械学習などの多くの知識処理技術を含んでいる。

3 応用プログラムの概要 (1)

3.1 論理シミュレータ

3.1.1 背景

論理シミュレーションは、設計回路の論理機能、および信号伝播タイミングの検証が行われる工程である。これは、LSI設計の中で最も計算時間を必要とする工程の一つであることから、高速化が強く望まれている。論理シミュレータの並列化は、高速シミュレーションを実現する有望な方法である。

並列論理シミュレーションは、並列離散事象シミュレーションの問題として扱われる。並列離散事象シミュレーションでは、複数のオブジェクトが互いにメッセージを交換し、その状態を変化させていくことでシミュレーションが進行するように問題をモデル化する。メッセージは事象情報をもち、その生起時刻が刻印されている(タイムスタンプ)。正しいシミュレーション結果を得るためには、各オブジェクトで時刻順にメッセージを処理していかなければならない。このため、通常は何らかの時刻管理機構が必要になる。

ICOTでは、分散メモリ型MIMD計算機であるPIM上で効率的に動作する論理シミュレータの構築を目指し、時刻管理機構としてタイムワープ機構を採用した。タイムワープ機構では、ロールバック処理が必要であり、そのオーバヘッドは大きいと予想されてきた。しかし、MIMD計算機上での実装例は極めて少なく、詳細な評価はほとんど行われていない。もし、ロールバックのオーバヘッドを小さく抑えることができれば、タイムワープ機構は分散メモリ型MIMD計算機に適した時刻管理機構になると考られる。このため、適切な負荷分散戦略を採用し、局所メッセージスケジューラ、アンチメッセージ削減機構を追加することで、効率的な並列論理シミュレーションの実現を試みた。

3.1.2 システム概要

本シミュレータは、組み合わせ回路、順序回路(非同期回路も含む)ともに扱うことができる。信号値モデルとしては、Hi、Lo、X(不定)を用いる3値モデルを、また、遅延値モデルとしては、各ゲートに異なる値を割り当てることができる非単一遅延モデルを採用した。なお、本シミュレータはゲートのみを扱うため、フリップフロップなど機能ブロックは全てゲートに分解して記述する。

本シミュレータでは、ゲートをオブジェクトとして表現している。タイムワープ機構 [Jefferson 1985] を用いた論理シミュレーションでは、各オブジェクトは時刻順にメッセージが到着するであろうという予測に基づき、履歴を保存しつつ、メッセージ処理を進める。もし、メッセージが遅れて到着した場合は履歴を巻き戻し(ロールバック)、そのメッセージが本来到着すべきであった時刻から処理をやり直す。また、誤って送信されてしまったメッセージに対しては、それらを取り消す役割を持つアンチメッセージを送り出す。

ロールバック処理のオーバヘッドは、無視できない大きさであるため、高速シミュレーション実現のためには、このオーバヘッドを削減する必要がある。また、PIMのような分散メモリ計算機を用いる場合、プロセッサ間通信オーバヘッドも小さく抑える必要がある。本シミュレータでは、縦割り指向戦略に基づく負荷分散、局所メッセージスケジューラ、およびアンチメッセージ削減機構によって、プロセッサ間通信およびロールバックのオーバヘッド削減を行っている。

- 縦割り指向回路分割

負荷分散を行う時の目標は、(1)負荷の均一化、(2)プロセッサ間通信低減、(3)高い並列性抽出の3点となる。本シミュレータの開発に際して提案、採用した縦割り指向戦略は、縦方向に連なったゲートを切り出す操作を基本とした静的回路分割戦略である(図3)。この戦略は、短い処理時間で上記3点の目標をある程度満たす手法と考えられる。

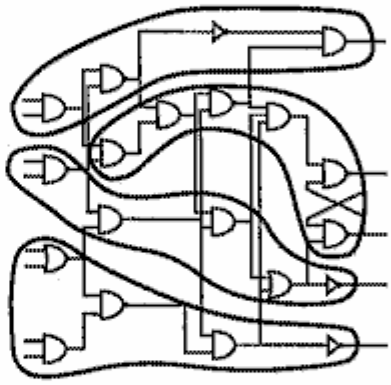


図 3: 縦割り指向回路分割

- 局所メッセージスケジューラ

通常、シミュレーション中には、一つのプロセッサに複数のメッセージが存在する。タイムワープ機構を用いた時には、これらのメッセージのうち、タイムスタンプが大きいものほど、巻き戻される可能性が大きいと考えられる。本シミュレータでは、メッセージ評価順を適切にスケジュールすることによってロールバック頻度を小さく抑えるようにしている。

- アンチメッセージ削減機構

KL1ストリーム通信では、メッセージは送信時と同じ順序で受信される。このような環境においては、連続して送信されるアンチメッセージは削減することができる [福井 1989]。メッセージ取り消しに必要なアンチメッセージ数が削減できることは、ロールバックコストの低減につながる。

3.1.3 計測結果

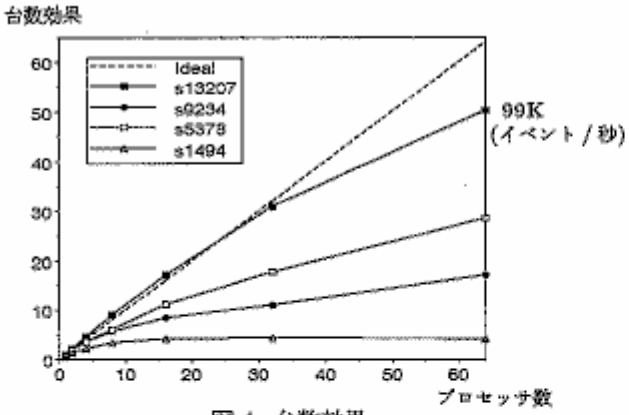


図 4: 台数効果

ISCAS'89 のベンチマークから 4 つの順序回路について、Multi-PSI 上でシミュレーションを行った。

図 4 に台数効果および最高性能値を示す。64 プロセッサを用いた場合、最大で 48 倍の台数効果と 99K イベント / 秒のシステム性能を得た。この台数効果は非常に良好であり、タイムワープ機構が効率良く動作したことを示す。システム性能についても、ソフトウェア論理シミュレータとしては良好な値であると考えられる。

なお、やや不満足な台数効果を示した回路については、その原因を明らかにするため、並列性解析および各種オーバヘッド計測を行った。その結果、問題の持つ並列性自身が小さいことが低い台数効果の主たる要因であることが判明した。詳細は文献 [Matsumoto *et al.* 1992] に記されている。

3.2 LSI レイアウト システム

3.2.1 背景と目的

LSI レイアウト処理は、二つのステージからなる。最初の配置処理では、回路を構成する部品の物理的位置が決定される。そして次の配線処理において、各部品における端子間の配線経路が決定される。これらの処理は、LSI 設計の中で最も時間を要する処理の一つである。故に、高性能なレイアウト CAD システムは、設計期間の大幅な短縮につながる。

我々の目的は、並列レイアウトプログラムの開発を通して、並列アルゴリズムや負荷分散方式について研究することである。また、マルチ PSI や PIM 等の分散メモリを持った MIMD 型並列計算機上に高性能なレイアウトシステムを構築することを目指している。

3.2.2 概要

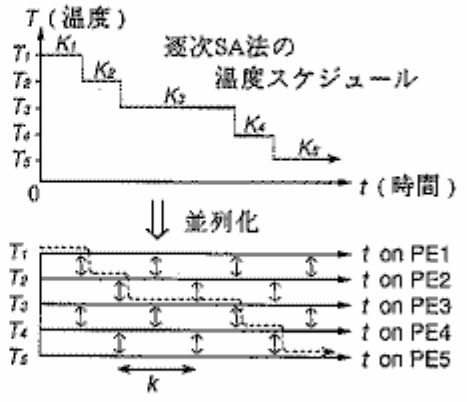
（1）配置システム　我々の配置システムは、マクロブロックを含まないスタンダードセル型の LSI を対象としている。スタンダードセルとは、高さが一様で、幅が可変なセルで、これらのセルは、チップサイズ (厳密には、総仮想配線長) が最小になるように複数のセルブロックに割り付けられる。このようなセル配置問題は、組み合わせ最適化問題として定式化できる。組み合わせ最適化問題を解く方法としてシミュレーテッドアニーリング (SA) 法が知られている。SA 法を効率的に実行する為には、温度スケジュールが重要である。そこで、我々の配置システムでは、適切な温度スケジュールを自動的に構成する時間的一様な並列 SA アルゴリズム [Kimura *et al.* 1991] を採用している。図 5 は、このアルゴリズムの概要を示している。

（2）配線システム　我々の配線システムでは、予測線分探索法 [Kitazawa 1985] に基づいて経路を探索する。このアルゴリズムは、高品質な配線経路を生成できるが、逐次アルゴリズムである。そこで我々は、配線問題を並列オブジェクトモデルに基づいて定式化し、上記のアルゴリズムを基に分散アルゴリズムを設計した。並列オブジェクトモデルは、細粒度の並列性を引き出すことが出来る。具体的には、配線格子上の各線分をオブジェクト＝プロセスに対応させた。これらのプロセスは、図 6 に示すように、各格子線に対応するプロセス (マスタ ライン プロセス) とその上の各線分に対応するプロセス (ライン プロセス) とからなる。格子線上にあるライン プロセスは、マスタ ライン プロセスを介して、直交するラインプロセスとメッセージを交換しながら配線経路を決定していく。

3.2.3 評価結果

（1）配置システム　我々のシステムを評価するデータとして、MCNC のベンチマークデータ (125 セル、147 ネット) を使用した。初期配置に関するエネルギー値は、911520 であった。また配線可能なチップ面積の下界を見積ると、1.372[mm^2] であった。

64 プロセッサを用いて 30 分実行させて得られたエネル



↕: 解の確率的交換, $1/k$: 交換の頻度

図 5: 時間的一様な並列シミュレーテッド アニーリング法

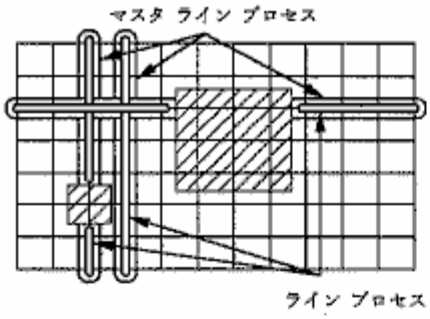


図 6: マスター ライン プロセスとライン プロセス

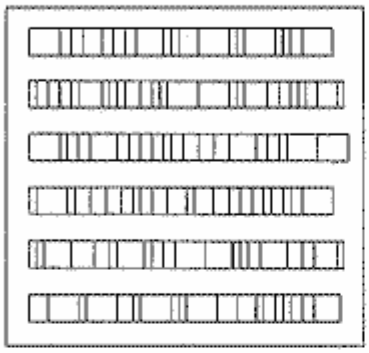


図 7: 配置結果

ギー値は、424478 であった。このときの配線可能なチップ面積の下界を見積もると、0.615 $[mm^2]$ であった。

これらの実験結果から、最終的なエネルギー値は、初期エネルギー値と比較して 56.0% 低下することが判った。

（２）配線システム　我々は、LSI の実データを用いて、次の三つの観点からシステムを評価した。(1) データ規模と台数効果との関係、(2) 並列性と配線率との関係、(3) 汎用計算機との性能比較。

図 8は、我々の配線プログラムをプロセッサ数を変化させて実行させたときの台数効果を表している。DATA2 は、DATA1 よりも大きなデータである。64 台のプロセッサを用いて最大 24 倍の台数効果を得た。

他の実験結果に関しては、[Date *et al.* 1992] に詳細に述べられている。

3.3　タンパク質配列解析プログラム

3.3.1　背景

タンパク質はアミノ酸が直線状に並んだものであり、これが生体内でおれたたまることにより、複雑な構造を形成する。

類縁関係にあるタンパク質を形成するアミノ酸配列が複数分かっている場合、それらの類似性解析を行う技術は、タンパク質の構造の予測、および進化系統樹の作成に役立つ非常に重要な技術であり、マルチプル・アライメントと呼ばれる。マルチプル・アライメント（以下アライメントと呼ぶ）は、下図のように、類似したアミノ酸同士を縦に並び合わせることにより行われる。アミノ酸は、慣習的に英文字一字で表されるため、同一のあるいは類似性の高いアミノ酸を表す英文字が、全体としてなるべく多く縦にならんでいるのが生物学的によいアライメントである。

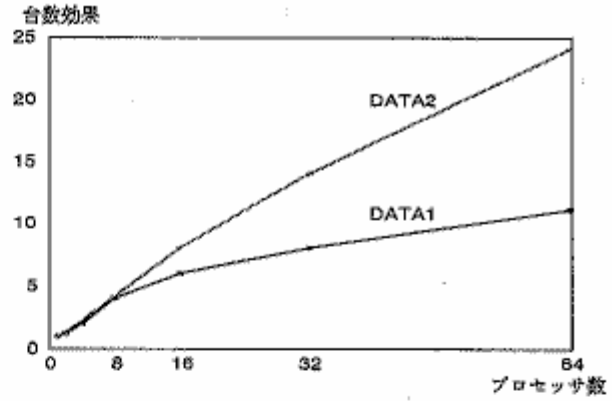


図 8: 台数効果

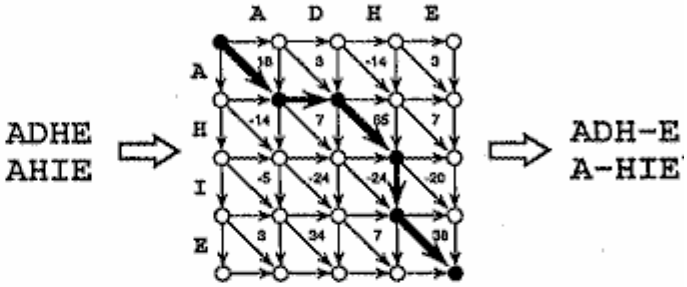


図 9: 2 次元 DP

```
...YICSFADCGAAYNKNWKLQAHLC-KH...
...FPCKEEGCEKGFTSLHHLTRHFL-TH...
...FTCDSDFCDLRFTTKANMKKHFNRFH...
```

以前は、アライメントは、生物学者の手作業で行われていた。しかし、既知のタンパク配列が増加するにつれて、計算機の利用が不可欠なものとなってきた。

アミノ酸同士の類似性尺度が与えられた場合、アライメント問題は、動的計画法（DP）で理論的には解決できることがニードルマンらの研究により、知られていた [Needleman *et al.* 1970]。DP によるアライメントは、アミノ酸配列に従って作られたメッシュ型ネットワーク上の最適経路を求める問題と等価である（図 9）。もし、N 本のアミノ酸配列をアライメントしたければ、理論的には N 次元の DP を行えばよい。

しかし、N 次元の DP によるアライメントは、配列の本数が多い場合、膨大な計算量を必要とするという問題点を持っている。N 本の配列をアライメントする場合、配列長の N 乗オーダの計算量が必要となる。現実的な時

問範囲内で、この問題を扱うために、今までに開発されたバートンらの方法 [Barton 1990] に代表されるほとんどのアライメント・システムは、2次元のDPの結果を何らかの方法で組み合わせることにより、3本以上のアライメントを行っていた。この種の方法は、時間がかからないという利点はあるものの、アミノ酸配列間の類似性が低い場合、質の高いアライメント結果を与えることができないという欠点を持つ。

3.3.2 プログラムの概要

質のよいアライメント結果を求め、なおかつ、計算時間の増大を少なくするために、いくつかのアライメント・システムを開発してきた。その内のひとつが、MASCOTと呼ばれる、DPをベースとするアライメント・システムである（図10） [広沢 1991]。

MASCOTは、タンパク質のアミノ酸配列が与えられた場合、まず、配列間の類似性に基づいて、配列を分類（クラスタリング）する。次に、それぞれのクラスタに対して近いものから順に3次元（配列3本）のDPを行う。3次元のDPを行うのは、2次元のDPに比べて質の高い結果が得られるためであるが、4次元以上は計算量の増大が大き過ぎるので行っていない。その後、各クラスタ内でシミュレーテッドアニーリング法という方法を用いて改善を行い、最後にそれぞれのクラスタをひとつにまとめる。

3.3.3 結果

MASCOTの各モジュールはKL1で記述されており、Multi-PSIあるいはPIM上で実行される。MASCOTは3次元DPを要素技術として用いているため、既存の方法に比べて計算量は増加しているが、並列推論マシンの並列処理が計算時間の短縮に貢献している [石川 1991]。図11は、3次元DPのスピードアップを示したものである。128台のプロセッサを用いて64倍の速度向上が達成されている。

MASCOTは生物学的に有用な結果を出力する。得られるアライメントでは、各クラスターごとに、配列上の重要なパターンを明確に整列させることができる。このようなパターン情報を、既存のモチーフ情報と比較しながらクラスター間アライメントを行うと、より品質の良い結果が得られそうである。現在、こうしたMASCOTの拡張を実現するために、知識工学的な手法も検討している。

3.4 折れ畳みシミュレーションプログラム

3.4.1 背景

折れ畳みシミュレーションは、タンパク質が、伸ばされた状態から、折り畳まれた天然状態までにいたる過程をコンピュータによりシミュレイトするものである。この研究分野は四半世紀に渡って生物学者により研究され続けている。なぜなら、タンパク質のアミノ酸配列順序を決めることは容易であり、タンパク質の構造を決定するのは困難（X線結晶解析やNMR (Nuclear Magnetic Resonance) により数か月）であるが、折れ畳みシミュレーションにより、タンパク質のアミノ酸配列順序よりタンパク質の構造を決定できるようになるからである。

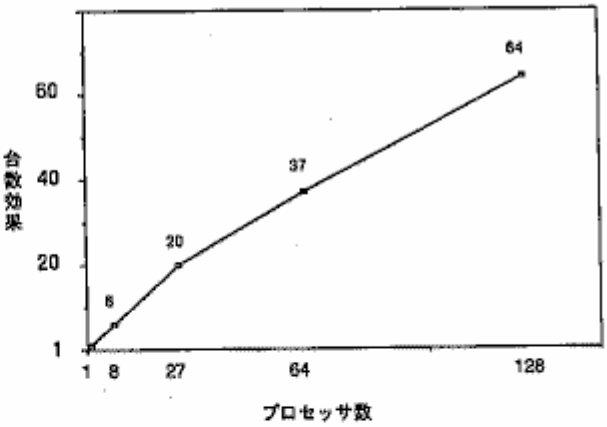


図 11: 3次元DPにおける速度向上

しかしながら、折れ畳みシミュレーションは莫大な計算時間を必要とし、現在のスーパーコンピュータを用いて解く場合にも近似を導入する必要がある。もっとも頻繁に用いられている近似が、タンパク質の格子表現である [Ueda *et al.* 1978] [Skolnick and Kolinsky 1991]。この近似ではタンパク質を構成するアミノ酸の位置を3次元格子点上のみに限定することにより計算量を削減している。

3.4.2 プログラムの概要

我々は、温度並列SA (Simulated Annealing)（図5の時間的一様な並列SAと同じ）を折れ畳みシミュレーションに適用した [Hirosawa *et al.* 1992]。折れ畳みシミュレーションを最適化問題として定式化するために、Water-counting model（格子表現（図12）を用い、アミノ酸間の相互作用として疎水性相互作用のみを考慮している）を導入した。このモデルでは、タンパク質が置かれていない格子点には、水が置かれている。

最小化するべきエネルギーは次の式で表される。

$$E(Energy) = \sum_m^{sidechains} (Water\ Count_m - 1) \times Hydrophobicity_m$$

$$Water\ Count_m = \frac{Number\ of\ adjacent\ cells\ (of\ side\ chain)\ occupied\ by\ other\ amino\ acids}{The\ number\ of\ adjacent\ cells\ of\ the\ side\ chain}$$

上記の式では、疎水性のアミノ酸（hydrophobicity が正）のまわりから水が減る場合、そして、親水性のタンパク質（hydrophobicity が負）のまわりに水が増える場

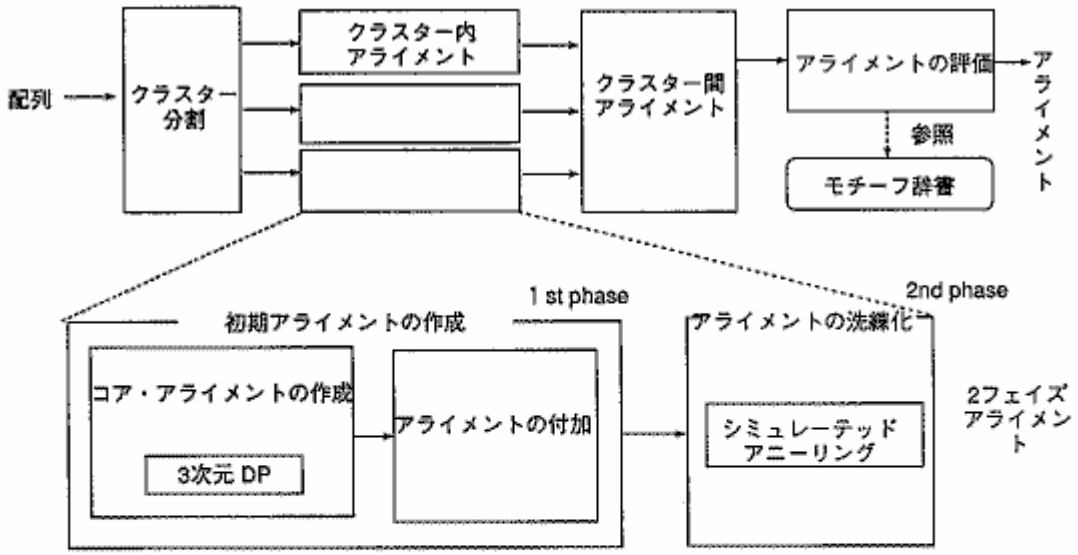


図 10: マルチプルアライメントシステム: MASCOT

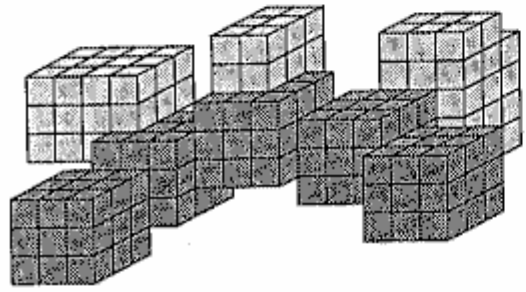


図 12: タンパク質の表現（一部）：タンパク質を構成しているアミノ酸は、主鎖（影付き）と側鎖（影なし）により構成される。

合に、エネルギーが減る。エネルギーを減らすことは、疎水性のアミノ酸をタンパク質の中心に呼びこみ、親水性のアミノ酸を周辺（水が多い）に追い出す効果がある。実際のタンパク質も疎水性のアミノ酸が中心にあり、親水性のタンパク質が周辺にあるので、このエネルギーを減少させることにより実際のタンパク質に近い構造を作ることができる。

3.4.3 結果

我々は、フラボドキシン（構造は知られている）をシミュレーションの対象とした。このタンパク質は１３８個のアミノ酸から構成され、中型のタンパク質である。我々は、２０個のプロセッサを用いた温度並列ＳＡによりシミュレーションを１０日間行った。これは、３００００サイクルに対応する。我々は、同様に２０個のプロセッサを用いた単純並列ＳＡによりシミュレーションを行った。

我々は、シミュレーションのエネルギー履歴（図 13）により以下の結論を導いた。

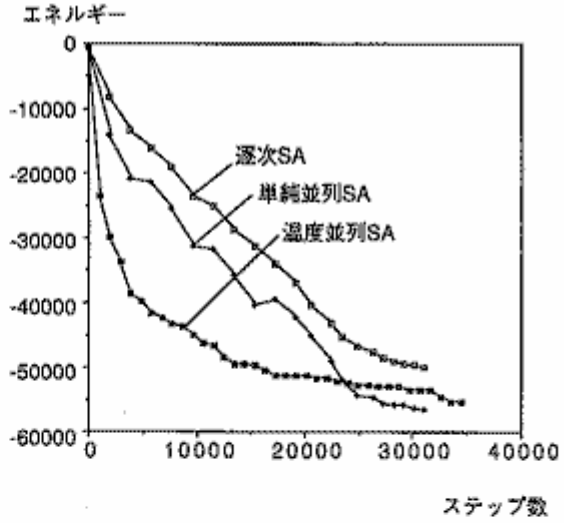


図 13: シミュレーションのエネルギー履歴

1. ２種類の並列ＳＡの結果は、逐次ＳＡより良い。これは、複数のプロセッサを用いた効果である。
2. シミュレーションの中頃まで、温度並列ＳＡは単純並列ＳＡより良い結果を示している。これは、温度並列ＳＡでは、シミュレーションのどの時点でもその時点における最適解を得られるからである。
3. 最終的に、２種類の並列ＳＡはほとんど同じエネルギーとなる。

3.5 タンパク質の構造解析プログラム

3.5.1 背景

タンパク質の立体構造予測は分子生物学の最も重要なテーマである。構造生物学者はこの問題を解決するためのさまざまな方法を提案してきたが、もっとも簡単に予測できると考えられる2次構造予測（特徴的な局所構造の予測）も実用的な精度で予測することは難しい。

3.5.2 プログラムの概要

我々はタンパク質の立体構造の予測方法を三つの段階に分けて開発を行うことにした。最初の段階で立体構造をできるだけ巧妙に表現する方法を開発する。次に第二段階では、その立体構造表現法によって表現された構造データとアミノ酸配列との関連を統計的に求め、その統計データから統計的に立体構造を予測する方法を開発する。さらに第三段階では、統計的に予測できた部分とできない部分を考慮し、予測できないところでは何が予測不可能の原因であるかを論理的に解析し、それから論理的な構造予測方法を開発するというものである。この第三段階に至ると、並列推論計算が重要なものとなり、並列推論マシンの特徴が有効に活用できるものと思われる。

現段階では、この三つの段階の最初の段階である立体構造の巧妙な表現方法の開発を行い、これまでには無い独自の表現方法として多変量解析を用いた表現法を考案した(図14)。

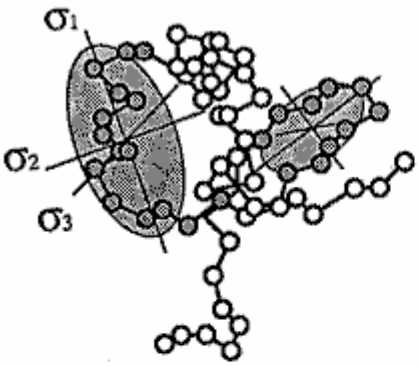


図 14: タンパク質のアミノ酸残基の空間分布の主軸

これは、タンパク質配列で連続するアミノ酸残基の立体構造での空間分布は、この分布の3本の主軸それぞれの上での分布の標準偏差で簡単に表現することが出来ることを利用したものである。

3.5.3 結果

この結果、多変量解析を用いることでタンパク質の立体構造の局所的な構造を数値表現することが可能であることが分かり、また、これまで提唱されてきた2次構造もこの方法で認識可能であることが分かった。更に、2次構造が複数個結合して形成される超2次構造の認識も可能であることが分かった。本システムによって得られた構造データは現状では数値データであるが、これを必要に応じて量子化することにより、論理的な解釈が可能な形にすることが可能である（図15）。

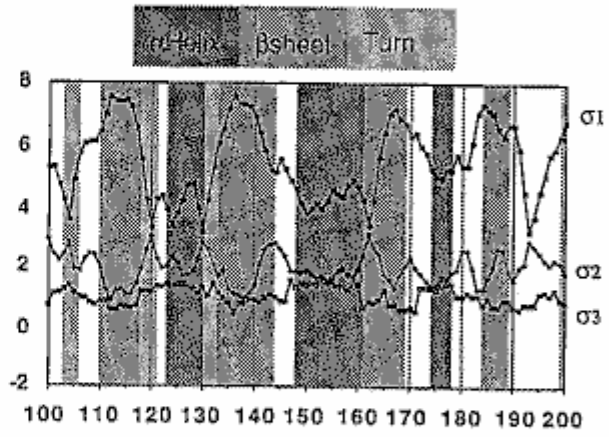


図 15: タンパク質のアミノ酸残基の配列に沿った空間分布

3.6 法的推論実験システム

3.6.1 背景

法律の主な知識は法令文と過去の判例からなる。法令文は法律ルールの集合であり、法令文に基づく推論はルールベース推論として実現できる。しかし法律ルールはしばしば定義の曖昧な法律用語（法的概念）を含んでいる。そうした法的概念の多くは、実際の事実に適用されて始めてその厳密な意味が決められるものである。このような法律ルールを実際の事実に適用するには、それらのルールを解釈することが必要である。すなわち、法的概念と具体的な事実との対応付けが必要になる。その際しばしば過去の判例が参照され、その中の論理展開が再利用される。つまり法的推論はルールベース推論と事例ベース推論の複合パラダイムとしてモデル化することができる。

実際に法的推論システムを開発するに際しては、幾つかの問題点がある。まず第1に法律ルール同様、過去の判例もまた非常に数多くあり、その中から類似の事例を検索し、それに基づいて結論を導き出すのには多くの時間が必要である。第2に複数の推論エンジンを管理するためには、それらの推論エンジンが行う推論を制御するための複雑な機構が必要となる。

我々は法的推論システムＨＥＬＩＣ－ＩＩを並列推論マシン上に開発し、これらの問題を並列推論によって解決した。

3.6.2 システムの概要

ＨＥＬＩＣ－ＩＩは与えられた事件に関する法的結論を、法令文と過去の判例を参照することによって導き出し、推論木の形でそれらを出力する [Nitta *et al.* 1992]。

ＨＥＬＩＣ－ＩＩはルールベース推論のエンジンと事例ベース推論のエンジンからなる（図16）。ルールベースエンジンは法律ルールを参照し、法的結論を論理的に導く。事例ベースエンジンは過去の類似の判例を参照して、与えられた事件から法的概念を導き出す。

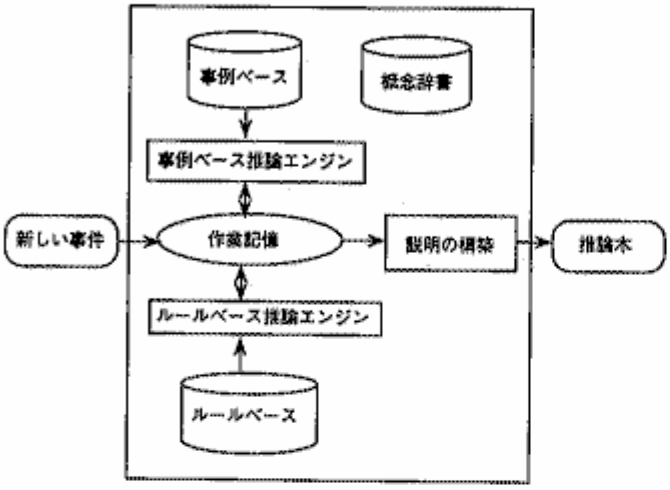


図 16: HELIC-II のアーキテクチャ

ルールベース推論　法的ルールは多数あるので、高速なルールベースエンジンが必要である。更に法的ルールには例外ルールも含まれている。ＨＥＬＩＣ－ＩＩのルールベースエンジンは並列定理証明器ＭＧＴＰ (Model Generation Theorem Prover)[Fujita *et al.* 1991] をベースにしている。ＭＧＴＰは非ホーン節の集合を与えると、全ての入力節を満たすモデルを並列推論によって生成する。

ＭＧＴＰを法律ルールのルールベースエンジンとして使うと同時に、パイプライン効果によって高い処理性能を得るために、我々はＭＧＴＰに幾つかの機能拡張を施した。

事例ベース推論　法的な事例（過去の判例）は弁護側と検察側の双方からの論証と、裁判官の判断、そして最終的な判決とから成る。我々は過去の判例を、その状況と幾つかの事例ルールの組として記述している。

状況は、事件の出来事に関する情報をイベントとオブジェクトの集合、そしてそれらの間の時間関係として記述する。双方からの論証は事例ルールとして記述される。

事例ベースエンジンは過去の類似の判例を参照して、法的な概念を生成する。このような機能を以下の２段階の処理によって実現している。まず第１段階では類似の判例を事例ベースから検索する。過去の判例は Multi PSI の各プロセッサに分配されており、それら過去の判例と新たな事件との間の類似度が各プロセッサ毎に並列に評価される。

第２段階では、選ばれた判例の事例ルールと、新たな事件との間の類似度を評価する。これにはＲＥＴＥネットに類似したネットを使っている（図17）。類似度の高い事例ルールを使って新たな事件に対する論証が構築される。

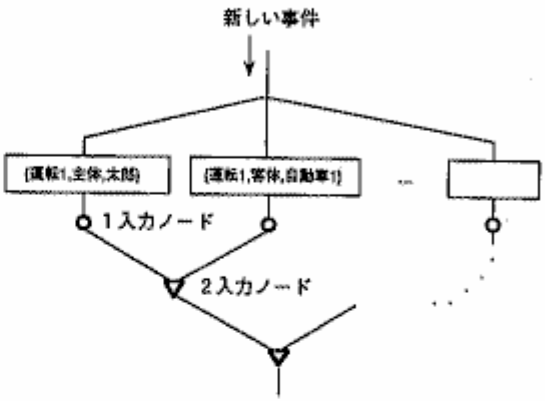


図 17: ＲＥＴＥネットに類似したネット

3.7 結果

我々はＨＥＬＩＣ－ＩＩが刑法の幾つかの事例を解けることを確認した。図18は事例ベースエンジンの第２段階での速度向上を示している。我々は Multi PSI の６４ＰＥを使って５０倍以上の速度向上を得ている。

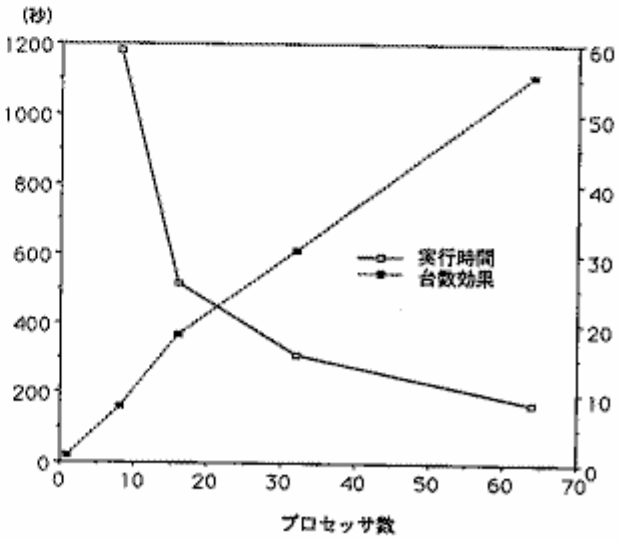


図 18: 事例ベースエンジンの性能

3.8 囲碁対局システム『碁世代(GOG)』

3.8.1 背景

囲碁は、日本・中国・韓国など東洋で人気のある伝統的なボードゲームである。囲碁は、黒石、白石、19 × 19の格子が描かれた碁盤を使って行われる。2人のプレーヤーは、交互に白石と黒石を格子の上に置いていき、相手よりも多くの陣地を取った方が勝ちというゲームである。囲碁は、完全情報ゲームである。

強い囲碁プログラムを作ることは難しく、未だ、アマチュア中級レベルの人間に勝つプログラムは存在しない。その難しさの主な原因は、(1) 平均的探索木の分枝数が多過ぎるために完全探索が不可能、(2) 簡単で正確な評価関数が存在しないという2つの理由による。

強い囲碁プログラムを作るためには、人工知能に関する各種の技術（探索問題、曖昧性の処理、例外処理、協調問題解決など）が要求されるので、知識処理の研究題材として適している。

我々は、並列推論マシンを使って、強い囲碁プログラム「碁世代」を開発中である。強さの目標は、アマチュアの中級レベルに達することとしている。

3.8.2 システム概要

碁世代には次の3つの特徴がある。

1. 人間プレーヤーの思考方法のシミュレーション
2. 大きな粒の処理の並列実行
3. 実時間性を保持しながら碁世代を強くするための新しい手法である「遊軍処理方式」

人間プレーヤーの思考方法のシミュレーション　碁世代の着手決定の手順は3つの段階からなる(図 19)。まず着手により変化した盤面上の石の生死などの局面の認識を行ない、それに基づいて候補手を列挙し、その候補手の評価値の合計がもっとも高い点の座標を次の着手位置とする。

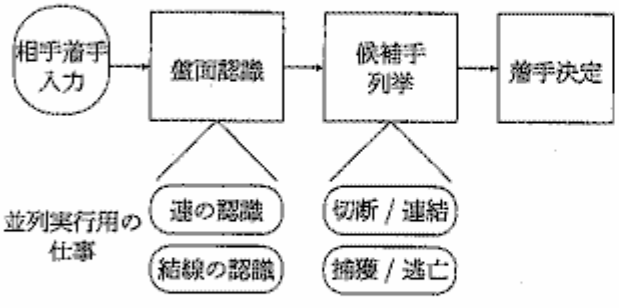


図 19: 並列版「碁世代」における処理の流れ

- 局面認識

  人間は石の配置を単に座標点の配置として捉えているわけではない。戦術的、戦略的に意味のある石の集団として捉えている。このシステムは、人間プレーヤーのように盤面上の単なる石の配置から、高いレベルのデータ構造を作り出し — 点、連（隣接する同色の石の極大集合）、群（近い位置にある同色の連の極大集合）、族（だいたいつながっていると思われる同色の群）、など —、それらの属性（ポテンシャル、地の大きさや形など）を算出する。

- 候補手列挙

  碁世代は、着手位置とその評価値を生み出す「候補手知識」を持っている。次の一手を決めるために、候補手知識から呼び起こされる処理の実行により、多くの候補手が挙げられる。碁世代が持っている候補手は12種類で、"定石"、"辺点"、"ダメ点"、"打ち込み"、"模様接点"、"捕獲 / 逃亡"、"連結 / 切断"、"包囲 / 脱出"などがある。

- 最終着手決定

  候補手は、各知識が互いに独立に挙げてくるため、異種ケース間の不調和を局所的に調整してより効果を高めたり、不調和を削除する。このような候補手間の調整を行った後で、各知識からの候補手の評価値の合計によって、最も評価値の高い点を打着する。

並列処理　並列版「碁世代」は、1つのマネージャプロセスと、多数のワーカプロセスとから構成される。マネージャプロセスは、ワーカプロセスで実行される仕事を作り出すこと、実行された結果から次の着手を決める処理を行う。ワーカプロセスは、マネージャプロセスによって渡された仕事の処理を行う。プロセッサへの割付けは、マルチPSIのプロセッサのうちの一つにマネージャプロセスを置き、残りのプロセッサにはワーカプロセスを一つずつ配置した。

システムが相手の着手を入力すると、局面の認識をし、候補手の生成をする。これらの処理において、大きな粒の処理である石の生死を判定する「捕獲探索」などを取りだし、ワーカープロセッサに割当てる。その結果がマネージャーに送られ、結果をもとに次の候補手を決定する。

遊軍処理方式　囲碁システムは、人間との対局を行なうため、短い時間内に着手を決めなければならない実時間性を要求されるシステムである。我々は、この実時間性を保持しながらシステムを強くすることを目的とした並列知識処理の一つのアプローチとして、「遊軍処理方式」を提案した。

遊軍処理方式とは、強くするためには重要ではあるが、次の着手には間に合わなくてもよい程度の処理を見つけ出し、それを「遊軍プロセス」に実行させるものである。遊軍処理を取り入れたシステムは、本軍と呼ばれるプロセス群と遊軍と呼ばれるプロセス群からなる(図 20)。本軍、遊軍ともに、一つのマネージャと、マネー

ジャから与えられた仕事を実行するワーカ群からなる。プロセッサ1番をマネージャプロセッサとし、本軍マネージャと遊軍マネージャが配置され、残りの各プロセッサには、本軍用ワーカプロセスと遊軍用ワーカプロセスが一つづつ配置される。本軍では、囲碁のルールに従って必ず行われる処理と、ある程度の強さを常に保つための処理を行う。

本軍は遊軍よりも高いプライオリティを持っている。着手を入力すると、本軍用マネージャプロセスは盤面認識や候補手列挙などの仕事を作り、本軍用ワーカプロセスに実行させる。遊軍で処理させる仕事は、遊軍用マネージャに送る。遊軍用マネージャは遊軍用の仕事を持っていないワーカプロセスに仕事を送る。ワーカプロセスは送られた仕事を終えると結果をマネージャプロセスに送る（ただし遊軍のプライオリティは低く、本軍用マネージャは全ての遊軍用ワーカプロセスの処理結果を待たないので、一般に遊軍用ワーカが仕事を受けとってから結果を返すまでの間に、何手か対局が進んでいる）。本軍プロセスは遊軍プロセスより高いプライオリティを持っているので、同じプロセッサに本軍用の仕事と遊軍用の仕事が与えられた場合、本軍用の仕事が先に実行される。本軍用の全ての仕事が終わると、本軍のマネージャプロセスは、その時までに終了している遊軍用の仕事の実行結果を集め、その結果と本軍によって実行された処理結果から、次の着手を決める。着手決定処理においては遊軍プロセスに投げた仕事の結果を待たないので、次の着手を生み出す時間は、遊軍に実行させる仕事には影響を受けず、遊軍用の仕事が増えてもシステムの実時間性は保たれる。

遊軍は次の着手を決める処理（本軍で行われる）とは独立に仕事を実行する。システムが着手した後、相手の考慮中も遊軍用ワーカプロセスは処理の実行を行う。また、遊軍が実行している仕事のもとになった盤面の部分が、対局が進むことによって変化し、実行中の遊軍の仕事が無意味になることがある。そのため、相手の着手を入力すると、遊軍用マネージャ内にあるまだ送っていない遊軍用の仕事が不要になるかどうかのチェックをする。また、既にワーカプロセスに送った仕事もチェックするために、各ワーカプロセスにも着手を伝える。

3.8.3 成果

表1に、碁世代の並列実行による台数効果を示す。この表から、序盤(30手目)・中盤(90手目)・終盤(180手目)のいずれの局面においても、並列実行によって思考時間が短くなったことがわかる。遊軍処理方式を取り入れた碁世代の強さについては、現在評価中である。

また我々は、逐次版の囲碁システム「碁世代」の開発も行っている。これは、盤面認識、候補手生成、着手決定段階における新しいアルゴリズムの実験を行うためである。昨年11月に行われた棋士システムワークショップの囲碁プログラムトーナメントに、この逐次版「碁世代」と、昨年の世界大会の1～5位のプログラムを含む7プログラムが参加した。碁世代の成績は2勝3敗で、世界のトップクラスに位置することを示した。人間のレベルでいうと、現在の碁世代は、初級者よりは強いが中級者よりはまだ弱い。

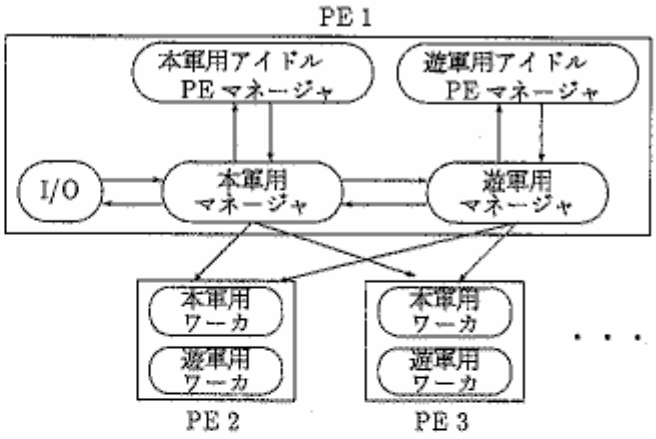


図 20: システム構成

表 1: 並列実行による台数効果 (単位: 倍)

第13期棋聖戦挑戦手合第1局			
局面	1 PE	4 PE	16 PE
30手目	1.0	3.3	5.1
90手目	1.0	3.4	5.3
180手目	1.0	3.7	7.5
第13期名人戦挑戦手合第5局			
局面	1 PE	4 PE	16 PE
30手目	1.0	3.2	5.4
90手目	1.0	3.4	5.6
180手目	1.0	3.6	5.9

4 応用プログラムの概要 (2)

4.1 Co-HLEX: 協調的再帰計算原理を用いたLSIレイアウト問題解決システム

LSIレイアウトは計算パワーを要するという点で最も巨大な問題の一つである。また、レイアウトシステムの開発と保守には多大なプログラマー労力を要する。この観点から、Co-HLEXの開発に当っては並列レイアウト算法とエレガントなプログラム記述との2点に関心を払った。部分問題が弱い連関を持つ問題に対しては古典的な分割統治原理が有効に働くが、本問題の場合にはこれが当てはまらない。チップ上の無駄な空きエリアを削減するためには、部分問題である隣接モジュール同士の形状や配線引き出し点が揃う必要が有る。FGCSパラダイムが提供する並列協調メカニズムはこの問題の解決に有効な手段を提供する可能性をもっている。

図 21にCo-HLEXの概要を示す。問題解決機能の核

部分は CMPN と呼ぶ4分木構造のプロセスネットワークである。レイアウト生成以前の CMPN の各ノードは、回路モジュール名、モジュール特性、他のモジュールとの接続ネット名、および配下回路モジュール名のリストからなる回路ネットデータを保持しているのみである。レイアウト後には、各ノードにレイアウトデータが追加される。その内容は、各ノード（の持つ矩形形状）を分割するのに使用したテンプレート (layoutframe) の名称、当ノードの持つ外部矩形形状、当ノード内の各ネットを配線するのに用いた配線パターン名称などである。HRCTL (Hierarchical Recursive Concurrent Theorem Prover for Layout) と呼ぶ算法を開発した。概要を示す。

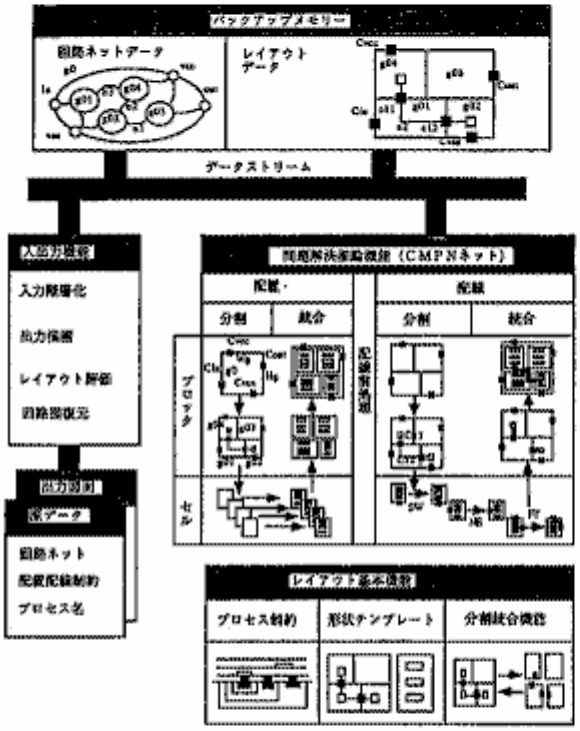


図 21: Co-HLEX の概要

配置 問題解決コーディネイタプロセスから CMPN の最上位ノードに対して配置実行メセージが送られる。メセージには、最上位ノードの目標形状と目標外部端子位置が含まれている。この後、CMPN の各ノードは再帰的配置行動を行う。最初のトップダウン相では、各非終端ノードは、別途用意されたテンプレートライブラリ内の適切な分割テンプレート (layoutframe) を用いて分割される。終端ノードでは、そのノードに対応したセルの幾何形状を定義する layoutframe が選択される。ボトムアップ相では子ノードのレイアウト形状が統合されて親ノードの形状が定まる。

配線 信号ネット配線との干渉を避けるために、最初に電源ネットである Vcc と Vee の非終端ネットを配線する。（CMPN の非終端ノードが保有するネットが非終端ネットであり、終端ノードであるセル内のネットが終端ネットである）。次に、信号ネットの非終端部分が配線される。CMPN の配線行動も再帰的である。CMPN の非終端ノードが保有する全てのネット外部端子存在範囲 (CERW：上位ノードから下達される) を狭めた後、各ネットについて内部区画間を結ぶための適切な配線パターンを選択する。配置時に選択した layoutframe には、固有な配線パターンが備えられており、この中から適切なものを選ぶ。選択したパターンが内部の区画境界線を通過する点に新たな配線端子が定義され、以降の下位ノードでの再帰処理において各内部区画の外部端子として使用する。

再帰は終端のセルノードで終結する。この時点で CERW はセルの高さあるいは幅の大きさまでに狭められる。非終端ネット配線の後、迷路法を用いた終端ネット配線、すなわちセル内部配線を行う。この際、全セルの内部端子から先ず SE（南東）、次に NW（北西）方向に配線を行い、最後に内部端子を持たない ND（非特定）方向という具合に配線方向をグローバルに管理し、配線位置決定用セル間通信量を低下させている。

約1000 モジュールからなるバイポーラ・アナログ回路を用いた実験を行った。緊密なモジュール配置と、不要な屈曲を持たない配線を生成できた。ランタイムでのモジュール間の協調的配線端子位置決定行為により配線用チャネルが不要となりチップ面積が削減される。

Co-HLEX の計算時間複雑度は約 $O(N)$ である。N はモジュール数。迷路法を用いたシステムでは約 $O(N^2)$ である。

Co-HLEX のプログラムサイズは KL1 で O(1000) であり、通常の 100,000 行にも及ぶ実現と対照的である。これは、HRCTL の再帰性と、KL1 のストリーム並列計算モデルによるモジュール化記述に負うところが大きい。

4.2 協調型論理設計エキスパートシステム

ＣＡＤシステムが抱える深刻な問題のひとつに、設計結果が制約条件をすべて満足するまで評価・再設計のサイクルを繰り返す手段がないことがある。このような手段なしでは、大域的な視点から所望の特性（面積及び動作速度）を備えた回路を設計することは不可能であろう。

co-LODEX は、仮説推論を用いた評価・再設計機構に基づくマルチプロセサ上の協調型論理設計エキスパートシステムである [Maruyama 1988][Maruyama 1990]。この機構では、設計における選択肢 (design alternative) を仮説と見なし、制約違反を矛盾とする。再設計は矛盾の解消として実現されている。中心的な役割を果たしているのは、制約違反の十分条件である制約違反条件 (NJ : Nogood Justification) である。co-LODEX は、設計すべき回路をあらかじめ部分回路に分割し、各設計エージェントが別々の部分回路を設計することによる並列処理を行う。エージェント間で設計結果や制約

違反条件を交換することにより、大域的な評価・再設計が行われる。他のエージェントから受け取った制約違反条件は、自分の探索空間を狭めるのに役立つ。受け取った制約違反条件を基に作る新しい制約違反条件により、探索空間の枝刈りができるからである [Maruyama 1991]。co-LODEX を「協調型」と呼ぶ理由はここにある。　co-LODEX は、ハードウェア記述言語で書かれた動作仕様、データパスに関するブロックダイアグラム、及び、面積と動作速度に関する制約条件を入力する。面積に関する制約条件はゲート数の不等式で、動作速度に関する制約条件は伝播遅延時間の不等式で、それぞれ表現される。co-LODEX は、これらの制約条件を満足する CMOS スタンダードセルの接続情報（ネットリスト）を出力する。このネットリストは、CMOS スタンダードセル用の自動配置・配線プログラムに入力可能である。

co-LODEX は、設計すべき回路全体を部分回路に分割し、それぞれを各エージェントが設計する。図 22 には、2 階微分方程式を解く回路（DiffEQ）の 5 つの部分回路と担当するエージェントを示す。

各エージェントは、与えられた機能ブロックをトップダウンに設計する。すなわち、機能ブロックを部分ブロックに、さらにその部分ブロックにと、CMOS スタンダードセルで直接実現できるまで分割を繰り返す。

そして、実現した回路を面積及び動作速度に関する制約条件とチェックするため、ゲート数を計算し、遅延時間を見積る。各エージェントは他のエージェントと並列かつ独立に設計を行っているが、大域的な制約条件に対するチェックを行うためには他のエージェントの設計結果が必要であり、設計／再設計の度にエージェント間でその結果を交換し合う。自分に責任のある制約違反を検出すると、エージェントは再設計を行う。

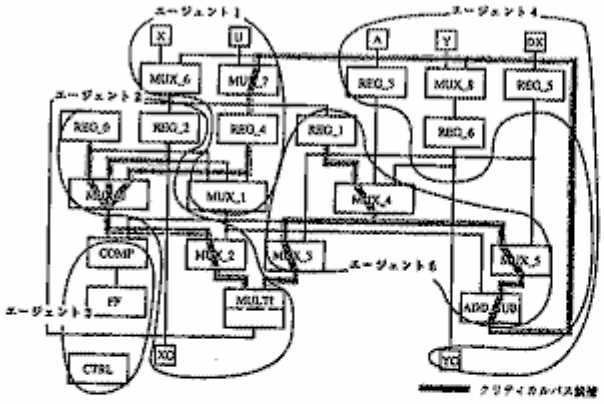


図 22: 部分回路とエージェント

我々は co-LODEX を KL1 でマルチＰＳＩ上に実装した [Minoda 1992]。実験結果によれば、co-LODEX は効率的に大域的な最適化を行うことができる。設計エージェントはプロセサに 1 対 1 に対応しているが、機能ブロックを各エージェントに分配したり統計データを取るため、特別なプロセサを 1 台設けている。図 23 には、均一性の高い回路に対する、エージェント数（1～15）と速度向上との関係を示す。

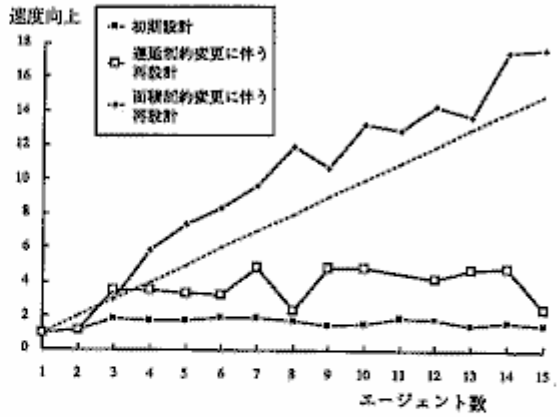


図 23: エージェント数と速度向上の関係

4.3　事例に基づく回路設計支援システム

最近、大量の知識を容易かつ効果的に獲得、利用するためのソフトウェア技術として事例ベース推論が注目されている。ここでは、デジタル回路の上流設計問題を通して、柔軟で高速な事例ベース推論機構に関する研究を行なった。

ここでは、入門書レベルの基本回路に関する知識は持っているが、設計経験の少ない初心者が教科書のレベルから少し外れた応用問題を解く場面を問題状況として想定している。従って、本システムでは、設計事例と基本回路に関する知識のみから、類似した既知の設計回路を検索し、検索された回路を修正しながら組み合わせることによって要求仕様に近い回路のブロック図を構成することになる。

本システムは、類似した機能構造を持つ回路の検索を行なう部分に特徴があり、検索のためのエンジンとして、構造写像エンジン（SME）[Falkenhainer 86] を利用している。構造写像エンジンは、上位レベルの関係構造が一致しておれば下位の構造に多少の違いがあっても、「構造的類似性」をもつと見做して、与えられた問題に対する類似事例を求めることができる。本システムの場合、構造写像エンジンにより、機能系統木間の類似度が評価される。すなわち、回路ブロックの基本機能とそれを実現するための副機能間の関係を階層的に木構造の形で記述したものの類似度が評価される。そして、細部の仕様は異なっていても全体的に似た機能を持つ回路が検索されてくる。例えば、時計回路に対して、事例ベースの中に時計回路そのものがなくても、回数や金額のカウンタ回路があればそれらが検索される。

本システムの構成を図 24 に示す。以下、設計プロセスの概略を述べる。

最初に Analyzer が要求仕様を分析して、データの流れに沿った複数の機能系統木と問題事例の詳細な表現を生成する。次に、ＳＭＥを用いた Retriever により、機能

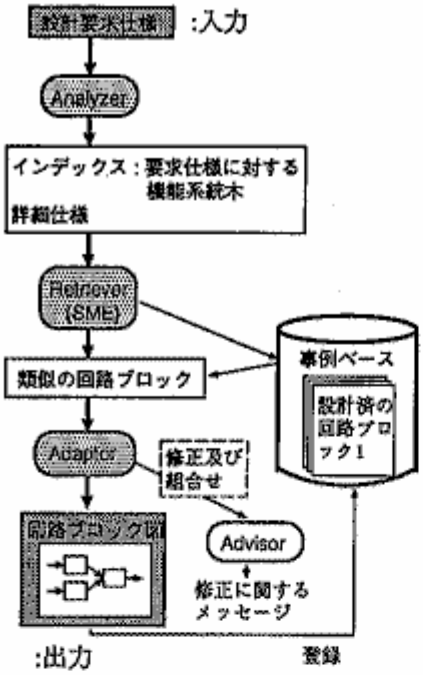


図 24: 事例に基づく回路設計システムの構成

構造が類似している設計事例を検索する。検索された類似の設計事例は Adaptor において詳細仕様が一致するかどうか検査され、一致しない場合は詳細仕様に合うように変更可能かどうかが調べられた上で、修正対応が可能であると確認された検索事例が組み合わせられる。このフェーズではまた、ＳＭＥによる失敗予測と回避の処理が行なわれるとともに、Advisor を通してその結果が出力される。最後に、生成された事例の組み合わせに対応するブロック図が表示される。出力ブロック構成図は使用者により評価され、適切であると判断された場合は事例ベースに新たな事例として登録される。

本アプローチにより、「温度検知機能付き時計」や「エアコンの性能計測装置」といった、決まりきった構成のものではないデジタル回路が実際に設計できることを確認した。 また、実験を通して、構造写像を利用した事例ベース推論方式の機能面での有効性を確認することができた。しかしながら、ＳＭＥは構造マッチングという組合せ問題を含むタスクを実行するため、処理の負荷が非常に重い。この問題に対しては、ＳＭＥのプログラムをマルチレベルロードバランサーを利用して並列化し、６４ＰＥ Multi PSI で１０数倍程度の台数効果が得られることを確認している。

4.4 並列ルールベース・アニーリングによる論理回路のハイレベル合成

図25はハイレベル合成 (HLS) の処理過程を示している。パスカルライクの言語 (Paspec) で記述された LSI の動作仕様を構文解析し、スケジュール表に変換する。スケジュール表は、各式がどの時間ステップでどの ALU で実行するかを示しており、生成する回路のデータパスに対応している。問題は、最もコストの少ないスケジュール表の式配置を見つけることである。コストは、チップ面積と実行速度の和として表せる。これは、組合せ最適化問題として解くことが可能である。

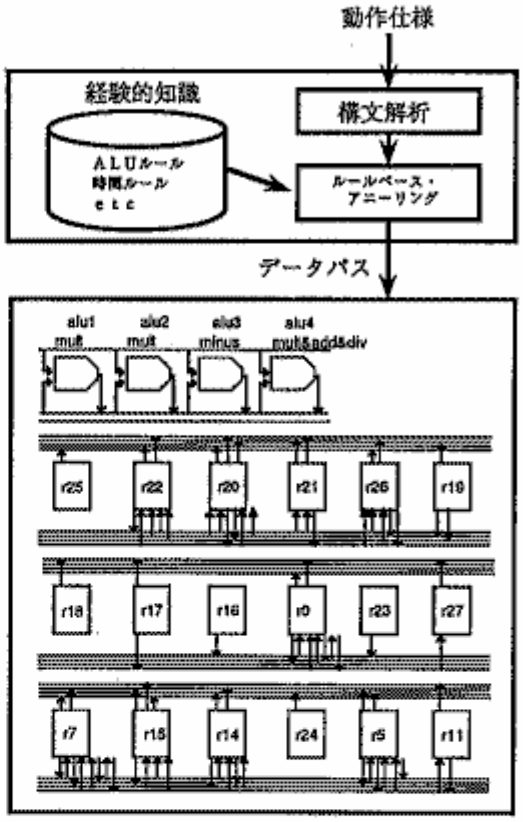


図 25: HLS の処理過程

並列ルールベース・アニーリング　シミュレーティッド・アニーリング (SA) は、組合せ最適化問題において最適解に近い解を得ることができるが、大量の変換回数が必要になる。また、ヒューリスティックなアルゴリズムは、より高速であるが、解が、極小解に陥りやすい。ルールベース・アニーリング (RA) は、それらの中間的な性質をもつように設計されている。各繰り返しステップにおいて、RA はランダムな変換だけでなく、ヒューリスティックな知識を利用した変換により次のスケジュール表の配置候補を生成する。ルールは確率的に選択され、各温度ごとにルールの選択確率は変化する。配置候補の受理率が高くなればなるほど、選択確率は高くなる。

我々は RA アルゴリズムの並列処理方式を設計した。システムは、１つのマスタプロセッサとその他のスレーブプロセッサから構成されている。各プロセッサは、同一温度で独立に RA を実行し、異なる配置を探索している。各温度のアニーリングの始めに、マスタプロセッサ

はコストに基づいてスレーブプロセッサを高いコストのグループと低いコストのグループに分割する。そして、2つのグループのコスト分布がほぼ等しくなるまで、アニーリングを実行し、次のアニーリングに移る。これは、高温度での無駄な変換を削除している。低温部では、極小値に陥ったと判断される配置は、その他のプロセッサのよい配置に置換される。

並列RAアルゴリズムは16プロセッサのMulti-PSI上にインプリメントされている。図26は、実験結果を示している。RAアルゴリズムは、SAアルゴリズムより4倍、並列RAアルゴリズムは、逐次RAアルゴリズムより、8倍の高速化をそれぞれ実現している。並列RAアルゴリズムの有効性を実験的に証明することができた。

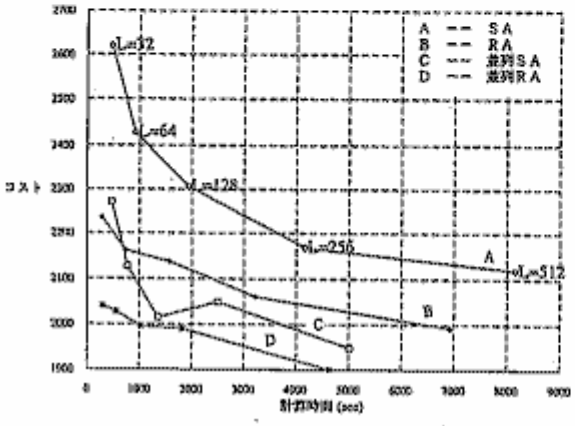


図 26: 実験結果

4.5 深い知識に基づく設計支援システム

設計においては、設計者が新規に装置を設計するのではなく、むしろこれまで設計した装置を変更したり改良したりすることが多い。しばしば、設計者は、仕様を満足するように、装置の構成要素のパラメータのみを変更する。そのような場合、設計者は、装置の構造はわかっているが、構成要素の新しい値を決める必要がある。電子回路では、このような設計は、頻繁に行なわれる。Desq (Design Supporting system based on qualitative reasoning) は、定性推論を使い、設計パラメータの妥当な範囲を求める。

Desqは、定性推論を使って対象とする系のすべての可能な振る舞いを求めるエンビジョニングを使用する。しかし、Desqの定性推論部は、量を定性的にも定量的にも扱える点で、通常の定性推論部と異なる。したがって、パラメータが定量的に与えられたならば、Desqは定量的な範囲を求めることができる。Desqのシステム構成を図27に示す。Desqは次の3つのサブシステムから構成される。

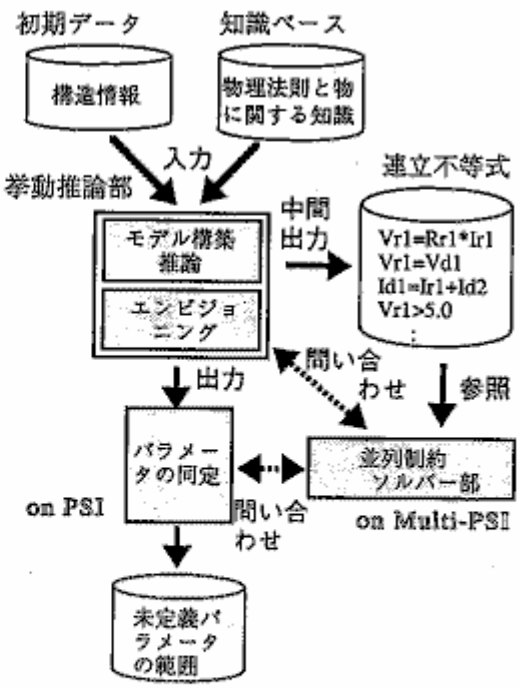


図 27: システム構成

挙動推論部

この推論部は定性推論部をベースにしている。そのモデル構築推論部は物理法則と物に関する知識を使って、初期データから連立不等式を作る。その連立不等式は対象とする系のモデルである。エンビジョニング部はすべての可能な振る舞いを求める。

設計パラメータ計算部

このサブシステムは、初期データの中で未定であった設計パラメータの値を計算する。

並列制約ソルバー

並列制約ソルバーは、連立不等式を解く。これはKL1で記述されており、並列推論マシンで実行することができる。

Desqは次のようにして設計パラメータの範囲を計算する。

(1) 初期データの中で未定である設計パラメータを基にしてエンビジョニングを行なう。

(2) エンビジョニングによって求められた可能な振る舞いの中から好ましい振る舞いを選ぶ。

(3) 好ましい振る舞いを与える設計パラメータの範囲を計算する。

一つの実験例として、Desqは図28に示すDTLの抵抗Rbの抵抗値の範囲を求めることができた。

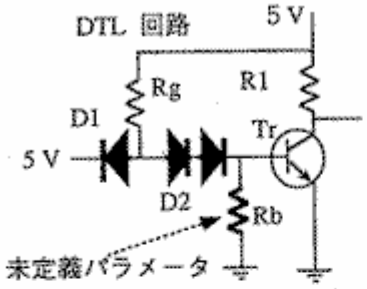


図 28: DTL 回路

4.6 プラントモデルに基づく診断制御エキスパートシステム

現在の火力発電プラントでは種々の高機能化が計られ、そのための運転制御知識も膨大になる状況にある。一般的な火力発電プラント運転制御システムにおいては、運転制御に関する経験則をいわゆる IF-THEN 型のルールに基づいて表現したものが多い。ところがこのような経験則に基づいた従来のシステムでは予め想定していない不測の事態に対処することができないという問題点がある。一方、経験則のように直接推論に利用する言わゆる浅い知識ではなく、浅い知識の正当性を証明しこれを生成できるような深い知識を利用した推論技術が診断分野を中心に研究されている。我々はプラント運転制御タスクにおいて、不測の事態を回避するための運転操作プランを自動的に生成できる深い知識を用いた推論機構について研究してきた。

本システムは、図 29に示すように浅い推論機構と深い推論機構から構成される。

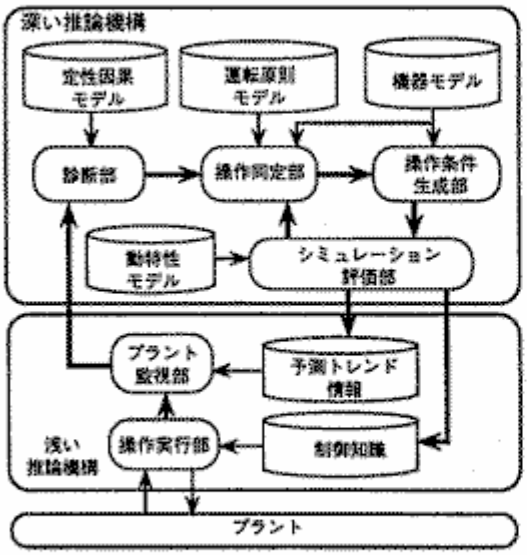


図 29: システム概要

浅い推論機構は経験則に基づいてプラント運転操作に関する推論を行う。プラント監視部は不測の事態の発生を検知し深い推論機構を起動する。

深い推論機構では、まず診断部が定性的因果モデルを利用して不測の事態の原因を推論する。操作同定部と操作条件生成部は、診断結果から不測の事態の回避に必要な運転操作プランを生成する。シミュレーション評価部は、生成された運転操作プランを検証するために、これが実行された時のプラント挙動を予測する。予測結果から、運転操作プランの修正が必要な場合は再び操作同定部を起動する。

以上の生成・検査手順を繰り返すことにより生成された運転操作プランは、浅い推論機構に送られ実行される。

本システムはマルチ PSI 上に KLI 言語により構築され、本格的な実験環境を実現するために、プラントシミュレータをミニコンピュータ G8050 上に構築し両計算機を GPIB 伝送路で結合した。

実際の火力発電プラントの機器（78 個）を対象に、機器モデルと動特性モデルを構築し、システムの性能を評価した。以下に結果を示す。

- 深い推論機構において、不測事態に対処するために必要な運転操作プランをダイナミックに生成できることが確認できた。
- 生成された運転操作プランを、浅い推論機構において実行することにより、不測事態に対処できることを確認した。
- マルチ PSI １６プロセッサを用いて推論時間の最大５倍の改善を実証した。

4.7 適応型モデルベース診断システム

従来の診断システムにおいては、症状と故障原因の関係をルール化して利用するルールベースの診断方式が多く用いられていた。しかしながら、この診断方式は、あまり柔軟ではなく、構築した知識ベースに入っていない「予期せぬ症状」が現れた場合に全く対応できないという問題点を抱えていた。これに対して、モデルベース診断と呼ばれる診断方式は、診断対象装置の構造や動作に関する知識を基に診断を行うため、より柔軟な診断が可能であるという長所を備えている。ただし、一般にモデルベース診断では、故障箇所を見つけるまでに要するテスト回数がルールベースの診断方式に比べて多くなるという問題点がある。これは、ルールベースの知識の中に、診断の専門家が持つ経験によるノウハウが活用されているためである。そこで、モデルベースの診断に対して、このような経験的な知識の学習機能を付け加えることで、経験を積むに従って診断効率を上げていくことが可能な「適応型診断システム」を開発した [Koseki *et al.* 1990]。

このシステムは、図 30に示すように複数のモジュールから構成されている。知識ベースとしては、「設計知識」と「経験的知識」の２種類の知識を用いている。設計知識は、診断対象装置が本来どのように動作するものであ

るかを記述したものである。同知識は、装置がどのような部品で構成され、各々がどのように接続されているかを示す構造的な知識と、各部品がどのような動作をするものであるかを示す動作知識とから構成されている。一方、経験的知識は、各部品の故障確率分布として表現される。診断部は、これらの知識を活用しながら診断を進めていく。

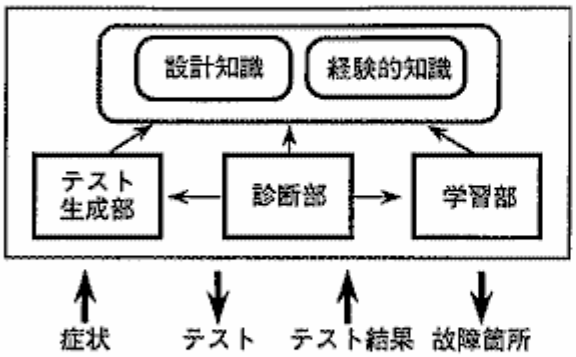


図 30: システム構成図

図31は、診断の流れを示した図である。診断の過程においては、被疑部品のリスト(SL)を持ち、新しい情報(症状、テスト結果)が入力される度に順次更新を行う。更新は、正常と判明した部品をSLから取り除く方法で行われる。まず、初期症状が与えられた時に、考え得る被疑部品のリストが作成される。ここで、症状は、入力値と(正常でない)出力値の組で表現される。このような入力に対してモデル(構造、動作に関する知識)を用いた診断計算が行われる。

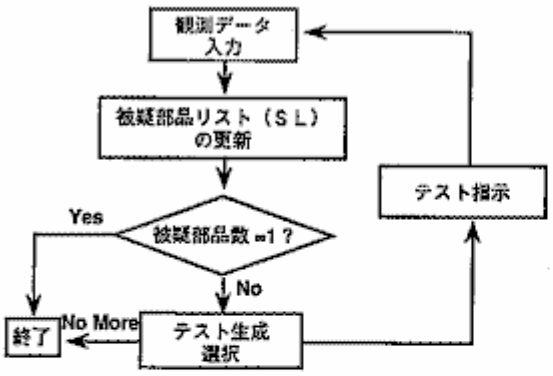


図 31: 診断の流れ

初期症状に対するSLを計算した後は、被疑部品を絞り込んでいくためにテストを生成しSLを更新するという動作を被疑部品が1個になるか、有効なテストがなくなるまで繰り返す。テストは、まず可能なテストの集合を生成部を用いて作成し、その中から最も効果的なテストが選択され、実行される。テストの効果は、そのテストによって得られる情報量とテスト実行に要するコストによって計算される。情報量は、各部品の故障確率によって計算される。

学習機能　テストの評価精度、つまり得られる情報量の期待値の推定精度は、各部品の故障確率の推定精度に依存するため、適切な推定を行うことが重要である。特に、過去の故障に関する情報が豊富に与えられないような状況では、偶然によるノイズを避けながら、有用な情報を引き出す必要がある。そこで、記述長最小(MDL)基準を用いて、与えられた情報に対して最も尤もらしい確率分布(確率モデル)の推定を行う方式を採用した。確率モデルの記述長は、モデルの複雑度を表すモデル記述長と、与えられたデータに対する対数尤度の和として計算される。この方式を用いることで、与えられたデータを最大限に生かして、将来起こる事象の確率分布の推定を適切に行うことが可能となる。

この学習アルゴリズムをKL1言語を用いてマルチPSI上にインプリメントした。実験の結果、16PEを用いることで約11倍の高速化が達成されることが確認された。

本適応型診断システムの学習効果について、学習の前後でどのくらい必要なテスト数が減らせるか、評価を行った。約70部品から成るパケット交換機を対象として実験を行った結果、学習前と比べて学習後には、平均で約40%程度効率が良くなる(テスト数を減らせる)ことが確認された。

4.8　モチーフ抽出システム

遺伝子情報処理の重要な課題の一つに、同種のタンパク質に共通なアミノ酸配列パターンの抽出がある。その共通パターンはモチーフと呼ばれ、タンパク質の機能／構造と密接な関係を持っている。

マルチＰＳＩ上に開発したモチーフ抽出システムを図32に示す。このシステムでは、モチーフを確率的決定述語で表現し、良いモチーフの探索に記述長最小（MDL）基準と遺伝的アルゴリズムを用いている。

確率的決定述語　記号パターンでモチーフを的確に表現することの困難さに対応するため、次のような確率的決定述語をモチーフ表現に導入した。

```
motif(S,cytochrome_c) with 129/225
   :- contain("CXXCH",S).
motif(S,others) with 8081/8084.
```

この例は、「もしSが"*CXXCH*"とマッチする部分列を含めば、Sは確率129/225でシトクロムｃであり、そうでなければ確率8081/8084で他のタンパク質である」ということを意味している。

記述長最小基準　記述長最小（MDL）基準は不確実性を含むサンプルデータに対して、過剰学習を避けながら良い確率モデルを推定する場合に有効な基準として知られている。MDL基準によれば最良の確率的決定述語は次の値を最小化するものであることが分かる。

述語の記述長 + 正確さの記述長

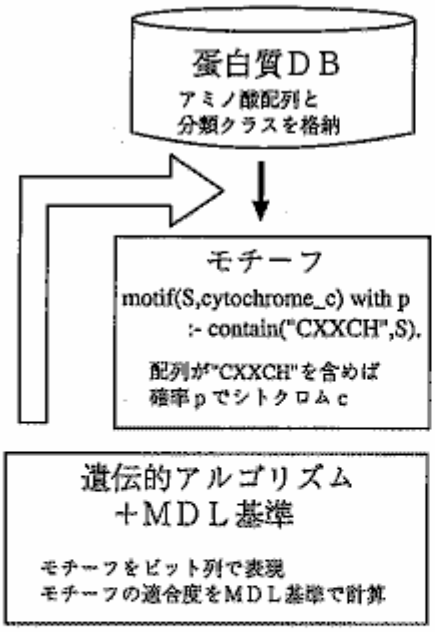


図 32: モチーフ抽出システム

述語の記述長は述語の複雑さ（値が小さいほど良い）を示し、正確さの記述長は述語のサンプルデータに対する尤度（値が小さいほど良い）を示している。それゆえ、MDL 基準はモチーフ表現の複雑さとサンプルデータに対する述語の尤度とのトレードオフを与えている。

遺伝的アルゴリズム　遺伝的アルゴリズムは進化を模擬した確率的探索アルゴリズムである。確率的モチーフは組合せ爆発を起こすだけの数があり全数探索で最良の確率的モチーフを見つけることは計算時間的に不可能であるため、遺伝的アルゴリズムを探索に採用した。

単純遺伝的アルゴリズムでは、関数 f の極値を以下の手順で探索する。

1. 関数 f の定義域の各点にビット列表現を与える
2. 複数個のビット列よりなる初期集団を生成する
3. 選択、交叉、突然変異の操作を用いて、繰り返し集団を更新する
4. 適当な世代を経た後、集団から最も良いビット列を得る

単純遺伝的アルゴリズムのモチーフ抽出への適用に当たっては、各ビットを一つのパターン（例えば、*"CXXCH"*）に対応させた 120 ビットのビット列をモチーフ表現に使用した。ビット列は、立っているビットに対応するパターンの連言を条件部に持つモチーフを表現している。

表 2はタンパク質配列データベース（NBRF）のシトクロムｃにモチーフ抽出システムを適用した結果である。同表において、抽出されたモチーフとその記述長が示されている。CL はモチーフの複雑さの記述長、PL は確率の記述長、DL はモチーフの正確さの記述長である。

表 2: シトクロムｃ

モチーフ	対象配列数	照合配列数	正例数
CXXCH	8309	225	129
others	8084	8084	8081

記述長 286.894
(CL = 16.288, PL = 10.397, DL = 260.209)

5　並列プログラムの性能解析

5.1　なぜ性能解析か?

各種応用プログラムの開発と平行して、より一般的な枠組の中で並列プログラムの性能の研究を行なってきた。主な関心は、大規模知識情報処理の問題を大規模並列推論マシン上で解く際の並列プログラムの性能である。

並列速度向上は、全体問題を数多くの部分問題に分割し、それらを並列に解くことに由来する。理想的には、並列化されたプログラムは p 台のプロセッサを用いて 1 台の場合と比べて p 倍の速度で解かれる。しかしながら、負荷の不均一、通信オーバヘッド、計算量の増大などのオーバヘッド要因が存在する。知識処理型のプログラムは、通常、(1) 部分問題の数やそれぞれの部分問題の大きさが予測不可能であること、(2) 部分問題間の通信パターンがランダムであること、(3) 全体の計算量が部分問題の実行順序に依存すること、などの点で「非定型的」である。このことにより、並列知識処理プログラムの設計においては、負荷バランス、通信の制御、およびスケジューリングが重要で、しかもトリビアルでない問題となっている。

プログラムを実際に走らせた時の実効性能はオーバヘッド要因により、マシンの「ピーク性能」と比べてはるかに悪くなる可能性がある。また、この性能差は単に定数倍（例えば、30% の性能低下）ではなく、プロセッサ数が増えるにしたがって増大する可能性がある。実際、設計の悪い並列プログラムでは、プロセッサ数が限りなく大きくなって行くに従って、ピーク性能に対する実効性能の比はゼロに近付く。

種々のオーバヘッド要因の振舞いを理解することができれば、並列プログラムを評価し、最も深刻なボトルネックを同定し、できればそれを除去することが可能となろう。究極の目標は、大規模並列推論マシンの適用可能性を広範な分野と問題に押し広げることにある。

5.2　初期の経験

並列推論マシン実験機マルチ PSI 上で稼働する最初のプログラムとして、比較的単純な問題を解く以下の 4 つのプログラムが開発された。これらのプログラムは FGCS'88 会議で実演されている [Ichiyoshi 1989]。

詰込みパズル（ペントミノ）

長方形の箱といろいろな形をしたピースが与えれていて、ピースで箱を埋め尽くす全ての仕方の求める

問題である。各ピースが5つの小正方形からなっているものはペントミノパズルとして知られる。プログラムはトップダウンなOR並列全解探索を行なう。

最短経路問題

各辺に非負のコストを持つグラフとその中の開始頂点とが与えられている時、開始頂点から残り全頂点への最小コストの経路を求める問題である（単一開始点最短経路問題）。プログラムは、分散グラフアルゴリズムを実行する。グラフとしては、正方形のグリッドグラフにランダム生成した辺のコストを付けたものを用いた。

自然言語構文解析

任意に与えられた英語の文の全ての可能な構文解析結果を列挙する問題である。プログラムは、ボトムアップチャートパーシングアルゴリズムを並列論理型言語で実現したPAXパーサ [Matsumoto 1987] である。チャートのエントリを表わすプロセスがチャートのデータフローを反映したメッセージストリームで結ばれている。

詰碁

碁の死活の問題である。プログラムはアルファベータ探索を行なう。

ペントミノプログラムでは、探索木の異なる部分を同時に探索することから並列性が生ずる。重なり合わない部分木は独立に探索できるので、部分タスク間には通信がなく、また見込み計算もない。そこで、負荷バランスが並列性能を決定する最大要因となる。最初のプログラムにおいて動的負荷バランス機構を実現し、64プロセッサで40倍以上の速度向上を得た。プログラムはあるプロセッサ（マスタプロセッサと呼ぶ）で実行開始し、探索木を展開し、探索部分タスクを生成し、暇な各ワーカプロセッサはマスタプロセッサに部分タスクを要求するという方式である。このプログラムにデータ構造やコードの改良を施したところ、逐次性能は向上したが速度向上は低下した。マスタプロセッサの部分タスク生成スループットがワーカプロセッサたちの部分タスク実行スループットに追い付かないことに原因があることが分かり、多重レベルの部分タスク割り当て方式を導入した。その結果、64プロセッサで50倍の速度向上を得た [Furuichi *et al.* 1990]。

この負荷バランス機構はプログラムから切り出され、ユーティリティとしてユーザに公開された。いくつかのプログラムがこれを用いたが、並列化した逐次深化A*アルゴリズムで15パズルを解くプログラムでは、ヒューリスティック関数による枝刈りのために探索木が非常にアンバランスであるにも係わらず、PIM/mにおいて128プロセッサで100倍以上の速度向上が得られている [和田ほか 1992]。

最短経路プログラムでは、隣接した頂点に対応するプロセスの間の通信が非常に多い。この場合、与えられたグリッドグラフの局所性を保つようなマッピングにより、プロセッサ間通信を抑えることができる。正方グラフをプロセッサ数分のメッシュに分割してそのまま割り付ける単純マッピングは局所性が一番高かった。しかし、計算が波面状に広がって行くために、実行中の各時点において忙しい状態のプロセッサは少数であり、結果として速度向上が低かった。グラフをより小さいピースに分割し、各プロセッサにグラフ内の互いに離れたいくつかのピースを割り当てることにより、プロセッサ稼働率が向上した [Wada and Ichiyoshi 1990]。

自然言語構文解析プログラムは、通信が多く、しかも通信が非局所的パターンであることが特徴である。最初の静的なプロセスマッピングでは速度向上が僅かしか出なかった。そこで、通信量を減らすために必要なデータのあるプロセッサにプロセスを移動させるように修正したところ、実行時間がほぼ半分となった [寿崎ほか 1989]。

詰碁プログラムは、ゲーム木のリーフまで並列アルファベータ探索を行なう。逐次アルファベータアルゴリズムは、最良ケースでゲーム木の実効分枝指数が半分になる。異なる枝を単に並列に探索すると、この枝刈効果の大部分を失うことになる。すなわち、並列版は無駄な見込み計算をしてしまう。詰碁プログラムでは、候補手探索タスクが候補手の見積り値に従って優先度付けすることにより、見込み計算量を減らした [沖ほか 1989]。

これらの初期のプログラムの開発を通じて、負荷バランス、通信の局所化、見込み計算量の削減などのいくつかの手法を開発することができた。

5.3 スケーラビリティ解析

並列実行オーバヘッドのより深い理解のためには、モデルを構築して解析することが必要である。この解析の結果は並列性能に対する洞察の土台となろう。

研究の中心は、並列プログラムのスケーラビリティである。大規模並列推論マシンを有効に活用できる良い並列プログラムは、性能がスケールするようなプログラムである。すなわち、プロセッサ数の増大に伴って性能が向上するようなプログラムである。例えば、2レベル負荷バランスは、1レベル負荷バランスと比べてより多くのプロセッサを稼働させられるので、よりスケーラブルと言える。しかし、どの程度スケーラブルかを決めるためには何らかの解析的な手法が必要である。

スケーラビリティの尺度として、Kumar と Rao の提案した等効率関数を採り上げた [Kumar *et al.* 1988]。問題インスタンスを固定すると並列アルゴリズムの効率（速度向上とプロセッサ台数に対する比率）はプロセッサ数が増大するにつれて低下する。多くの場合、効率は問題の大きさを増大させることによって回復できる。プロセッサ数 p を増大させる時に、一定の効率 E を維持するために問題サイズ（逐次実行時間で計る）が関数 $f(p)$ に従って増大しなければならない場合、$f(p)$ を等効率関数であると定義する。等効率関数は p の増加につれて最小限線形に増大する必要がある（さもないと、各プロセッサに割り当てられる部分タスクのサイズがゼロに近

付いてしまう）。種々のオーバヘッドのために、等効率関数の増加速度は一般に線形より真に大きいことが必要である。$p\log p$ のような緩やかな増加速度は、期待する効率を得るのに比較的に小さな問題を実行すれば十分であることを意味する。一方、指数関数のような非常に大きな増加速度は、実際的な規模の問題を実行することによっては、大規模並列マシンを有効に活用できないことを示す。

最初に、多重レベル負荷バランス方式を解析の対象として選んだ。確率的モデルと対象問題の持つべき性質についての仮定に基づいて、単一レベル負荷バランスと多重レベル負荷バランスの等効率関数が求められた。決定的なケース（全ての部分タスクが等しい実行時間を持つ場合）では、単一レベル負荷バランスの等効率関数のオーダは p^2 に対し2レベル負荷バランスでは $p^{3/2}$ であり、多重レベルによるスケーラビリティの向上の度合が理論的に分かったことになる。部分タスクサイズのバラツキに等効率関数がどの程度依存するかについても調べられ、部分タスクサイズが指数分布する場合、単一レベル負荷バランスでは $\log p$、2レベル負荷バランスでは $(\log p)^{3/2}$ の因子が掛かることが分かった。詳細は [Kimura *et al.* 1991] を参照のこと。

より最近になって、静的負荷分散の例として、分散ハッシュ表の負荷バランスについて研究を行なった。分散ハッシュ表とは、逐次ハッシュ表を等しいサイズの部分表に分割し、プロセッサに割り当てることによって並列化したものであり、ハッシュ表に対する多数の探索操作を並列に行なうことで、スループットを高めるものである。オーバヘッドは、主に、負荷不均一と通信オーバヘッドである。各プロセッサに割り当てるバケット数（部分表サイズ）を増加させることによって負荷バランスが改善すると期待される。よい負荷バランスを維持するために必要な部分表サイズの増加率の程度を調べた。非常に単純な負荷分散モデルを定義して解析し、負荷バランスに関する等効率関数を得た [Ichiyoshi *et al.* 1992]。その結果、平均負荷バランスが完全なバランスに近付くために必要な部分表サイズ q の増加率が、比較的緩やか（$q = \omega((\log p)^2)$）で十分であることが確かめられた。これは、分散ハッシュ表は穏当な大きさの問題を使って高並列計算機の計算力を引き出すことができることを示唆している。

5.4 残された課題

上記のように、大規模並列計算機の計算力を活用するためのいくつかの手法を確かめ、また、動的および静的負荷バランスそれぞれの実例についてスケーラビリティ解析を行なった。種々の並列化オーバヘッドの解析およびそれらの漸近的な性質を理解することにより、大規模並列処理に対するより深い洞察に導き、大規模並列マシン上のプログラムの設計の指針を与えることができる。

しかしながら、これまでの成果は大規模並列計算の新しい世界への探検のほんの入口に過ぎない。例えば、通信オーバヘッドや見込み計算を扱えるように解析手法を拡張する必要がある。数百プロセッサ構成の並列推論マシンが稼働開始した今、漸近的解析の結果を実験データと突き合わせて、適用度を評価することができよう。

表 3: プログラムのサイズと開発人月

プログラム名	サイズ	人月
論理シミュレータ	8 k	3
配置		
(KL1)	4 k	4
(ESP†)	8 k	4
配線	4.9 k	2
3次元 DP によるアライメント	7.5 k	4
SA によるアライメント	3.7 k	2
折り畳みシミュレーション	13.7 k	5
法的推論		
(ルールベースエンジン)	2.5 k	3
(事例ベースエンジン)	2 k	6
基盤代	11 k	10

†: システムプログラミング向けに拡張された Prolog

6 並列プログラム開発のまとめ

われわれは並列応用プログラムの概要と性能解析の結果を紹介してきた。ここでは PIM/KL1 による並列処理と知識処理をまとめる。

(1) PIM/KL1 による知識処理

CAD や診断や制御やゲームなどの並列応用システムの開発で用いられた知識処理技術はそれ自身で先端的なものであり、知識処理の観点からも価値が高いものである。これらの技術は計算時間が非常にかかるものが多く、逐次マシンでは大きな問題を解決するのが不可能な場合がある。従って、並列推論マシンの有効性を確認するには適当なテーマであった。

われわれはすでに逐次型の論理型言語で知識処理のシステムを開発した経験があるため、KL1 言語に慣れるのにあまり長い時間は必要なかった。一般的に、並列プログラムを開発するにはプログラマは各モジュールの同期を考慮しなければならず、それがしばしばプログラムミスの原因となっていた。KL1 では自動的な同期機構が言語自体に埋め込まれているため、表3のように比較的短時間で並列プログラムを開発することができた。

並列プログラムの効率が良くないときには、並列処理の単位を考え直す必要がしばしばあった。そのため、問題解決モデルの設計には逐次マシンよりももっと時間をかける必要があった。

(2) 2つのタイプのプロセスプログラミング

KL1 のプログラミングスタイルは逐次の論理型言語のスタイルと異なっている。KL1 の典型的なプログラミ

ングスタイルはプロセスプログラミングである。プロセスとは、内部状態とそれを操作する手続きからなるオブジェクトであり、プロセスの間はそれらを結ぶストリームを介して通信する。プロセスからなる構造は KL1 で簡単に実現することができ、また、多くの問題解決モデルはプロセス構造として表現することができる。

応用プログラムに現れるプロセス構造は大きく 2 つに分けることが可能である。

1. 静的なプロセス構造
   与えられた問題に対し、まずプロセスのネットワークが作られ、その中で情報が交換されながら問題が解決される。このプロセス構造は問題が解決されるまでほとんど変化しない。多くの分散アルゴリズムは静的プロセス構造である。並列応用プログラムの多くはこのタイプに属する。例えば、論理シミュレータにおいて、回路はいくつかの部分回路に分割され、それぞれがプロセスとして表現されていた（図３）。タンパク質の配列解析では２つの配列が KL1 プロセスの２次元ネットワークとして表現された（図９）。法的推論システムでは事例ルールの条件部はプロセスのネットワークとして表現された（図１７）。co-LODEX では個々の設計エージェントがプロセッサにわりあてられた（図２２）。
2. 動的なプロセス構造
   問題解決の途中でプロセス構造が動的に変化するタイプである。例えば、トップレベルのプロセスが子プロセスを生成し、それがまた孫プロセスを生成するような場合であり（図 33）、探索木がこのタイプに属する。従って、ペントミノや１５パズルや詰め碁はこのようなプロセス構造をとる。

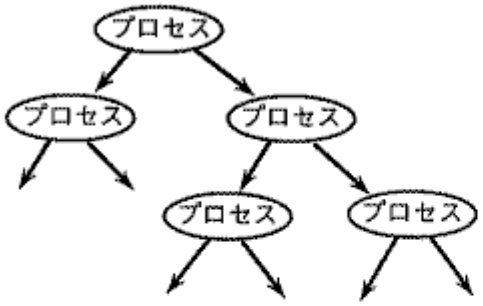


図 33: 動的プロセス構造による探索木

(3) 並列アルゴリズムの新しいスタイル

応用プログラムの開発において、単に逐次アルゴリズムを並列に書き直しただけでなく、元のアルゴリズムになかった好ましい性質を持った並列アルゴリズムがいくつか開発されている。

組合せ問題では、LSI 配置や遺伝子解析の MASCOT で使われた並列シミュレーテッドアニーリング (SA) や高レベル合成で用いられた並列ルールベースアニーリング (RA)、モチーフ抽出プログラムで使われた遺伝的アルゴリズム (GA) などがその例である。

並列 SA アルゴリズムは逐次 SA アルゴリズムの単なる並列版ではない。あらかじめ温度パラメータをプロセッサに割り当てておき、中途解がプロセッサ間で交換されることによって、評価値の高い解が温度の低いプロセッサに次第に集まるのである。このため、温度スケジュールの調整が不要になるという大きな利点がある。また、並列 SA アルゴリズムは時間軸について一様である。逐次 SA では極小に陥って抜け出せなくなることがありうるが、並列 SA ではその問題はない。

並列 RA アルゴリズムでは評価値の分散がモニタされ、定常状態に達したかどうかの判定に使われる。

囲碁システムでは、実時間問題解決に適した手法として遊軍処理が導入された。遊軍は地の大きな変動（大石の死活や大きな地の侵略など）を起こすかもしれない着手の結果を調べる役割を持っている。このような可能性のチェックは実時間システムでは許されないような長い時間がかかるので、遊軍でないと実行することはできない。

(4) 並列推論の効果

並列応用プログラムのいくつかでは 128 のプロセッサを用いて 100 倍以上の高い台数効果を達成した。例えば、論理シミュレータ (LS)（図４）、法的推論 (LR)（図１８）、第５研究室で開発された並列定理証明 MGTP [Fujita *et al.* 1991] [Hasegawa *et al.* 1992] がその例である。これらのプログラムでは、並列性があって、並列化にともなうオーバーヘッドができる限り抑制されていることが特徴である。論理シミュレータ（LS）も法的推論（LR）も MGTP も、その並列性は大きいデータ量（論理シミュレータではゲート数、法的推論では判例数）や問題空間の広さ（MGTP）にある。負荷のバランスは、プロセッサに均等にあらかじめ割りつける (LS, LR) か、動的に実行中に割りつける (MGTP) ことによる。通信の局所性は、プロセスをクラスタリングしたり（LS）、部分タスク間の通信がもともと少なかったり（LR におけるルール集合の分割、MGTP における OR 並列探索）することによって保たれている。

(5) 負荷分散パラダイム

これまでに述べた全ての応用プログラムにおいて、静的プロセス構造を持つものは静的負荷分散を用い、動的プロセス構造を持つものは動的負荷分散を用いている。

静的なプロセス構造を持つプログラムにおいては、各プロセッサにほぼ同数のプロセスを割り当てることによって、通常、良い負荷バランスを得ることができる。通信オーバヘッドを抑制するためには、論理的プロセス構造における局所性を保つことが望ましい。すなわち、まず、プロセス全体を、互いに近接したプロセスからなるクラスタに分割し、次に、それらのクラスタをプロセッサに割り当てる。実行途中においてプロセスの一部のみ

が高いレベルの計算活動を行なっているような場合、この直接的なクラスタからプロセッサへのマッピングはよい負荷バランスを得られない。そのような場合、プロセス構造をより小さいクラスタに分割して、互い同士離れたいくつかのクラスタを一つのプロセッサに割り当てることで負荷バランスが改善する。この多重マッピング方式は、最短経路問題および論理シミュレータで採用されている。3次元DPマッチングプログラムでは、アライメント問題（3つのタンパク質の配列）が次々にマシンに投入され、アライメントがパイプライン的に処理されることで、全てのプロセッサを忙しく保っている。

動的プロセス構造を持つプログラムでは、新たに生まれたプロセスを計算負荷の軽いプロセッサに割り当てることで、負荷をバランスさせることができる。通信オーバヘッドを低く保つためには、負荷分散の候補として少数のプロセスのみを選ぶ。例えば、木探索問題では、全ての部分探索タスクではなく、一定の探索深さの部分探索タスクのみをプロセッサ間負荷割り付け対象とした。ペントミノプログラムや15パズルプログラム、囲碁プログラムは、この要求駆動型動的負荷バランスを用いている。

(6) 並列処理の粒度

並列処理で高い性能を得るためには、並列処理の粒度を考慮する必要がある。各部分タスクのサイズが小さいとプロセススイッチや通信などのオーバヘッドが深刻になるために、高い性能を得にくい。例えば、論理シミュレータの最初の版では、回路の各ゲートが互いに通信し合うプロセスとして実現されていた。各プロセスの計算の量が小さいためにこの版の性能は高くなかった。次の版では部分回路ひとまとまりをプロセスとして実現し（図3）、性能改善につながった。

(7) プログラミング環境

マルチPSI上の初期のプログラムは、マシン上のKL1処理系が完成する前に開発された。ユーザは汎用機上のPDSS逐次処理系（PIMOS development support system）でプログラムを書き、デバッグを行なった後、負荷分散プラグマを付加してプログラムをマルチPSIに移植した。マルチPSI上のデバッグ機能は処理系自身のデバッグを目的としたもので、応用プログラムのデバッグは容易でなかった。その後、徐々にPIMOSオペレーティングシステム [Chikayama 1992] は、対話型トレース／スパイ機能、単一出現変数（単なるスペルミスであることが多い）に警告を出す静的なコードチェッカ、デッドロック報告機能などのデバッグ機能を備えて行った。このデッドロック報告機能は、永久中断したゴール群を同定し、（非常に多数かも知れない）その全てを出力する代わりに、データフローにおける最も上流のゴール（の一つ）を表示する。この機能は永久中断の原因箇所の発見に非常に役立っている（多くの場合、報告されたゴールの待っている変数を具体化しそこなったプロセスが犯人である）。

その後に、性能モニタ／収集機能が追加された（現在も機能強化されつつある）[Aikawa 1992]。実行後にプロセッサ稼働率を時間軸に沿って表示する機能は、実行中のあるフェーズにおいて特定のプロセッサがボトルネックになっていることを明白に示すことがしばしばある。プロセッサ時間の内訳表示（計算／通信／アイドル）によって、ボトルネック除去のためにどのようにプロセス構造を変えたらよいかのヒントが得られる場合も多い。

性能モニタ／収集機能からの情報を解釈して、性能をチューンするために、KL1処理系の知識が必要なことが時々ある。応用プログラムの性能チューンにおける同様の状況はどんな計算機においても存在するが、KL1のような並列記号処理言語においてはより深刻であるように思われる。ユーザの KL1 に対するイメージと処理系のギャップをどのように埋めるかは性能デバッグ／チューニングの今後の課題である。

7 まとめ

並列応用プログラムの概要と性能解析の結果について紹介した。ここに紹介した応用プログラムは並列処理の観点からも知識処理の観点からも興味ある技術を含んでいる。

KL1でいろいろな知識処理技術を開発し、PIM上での並列推論の効果を測定することによって、われわれはPIMとKL1が並列知識処理を実現するのに適したツールであることを実証した。また、われわれは、高い効率を得るための多くの並列プログラミング技術を開発し、その効果を実機で実験することができた。これらの経験はより大きな応用プログラムの開発の指針としてまとめられた。

性能解析チームは並列プログラムの動作を一般的な枠組として解析した。その成果はいろいろなプログラミング技術を選択したり、問題の大きさをスケールアップしたときの性能の予測をしたりするのに有用な情報を与えるものである。

この論文で示したデータはMulti-PSIやPIM/mによるものである。今後は異なったPIMのモデルの上の効率を測定し、比較することが必要である。また、多レベル負荷バランサの他に動的負荷バランサなどの並列プログラム開発のためのユーティリティの開発も必要である。

謝辞

並列応用プログラムの開発は、ICOT第7研究室の方々、関連のコンピュータメーカーの方々、ワーキンググループの方々のご協力によるものです。ここに深く感謝いたします。また、ICOT研究所 渕一博所長、内田俊一研究部長のご支援に感謝いたします。

参考文献

[Aikawa 1992] S. Aikawa, K. Mayumi, H. Kubo, F. Matsuzawa. ParaGraph: A Graphical Tuning Tool for

Multiprocessor Ssytems. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1992*, ICOT, Tokyo, 1992.

[Barton 1990] J. G. Barton, Protein Multiple Alignment and Flexible Pattern Matching. In *Methods in Enzymology, Vol.183* (1990), Academic Press, pp. 626-645.

[Chikayama 1992] Takashi Chikayama. KL1 and PIMOS. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1992*, ICOT, Tokyo, 1992.

[Date *et al.* 1992] H. Date, Y. Matsumoto, K. Kimura, K. Taki, H. Kato and M. Hoshi. LSI-CAD Programs on Parallel Inference Machine. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1992*, ICOT, Tokyo, 1992.

[de Kleer 1986] J. de Kleer. An Assumption-Based Truth Maintenance System, Artificial Intelligence 28, (1986), pp.127-162.

[Doyle 1979] J. Doyle. A Truth Maintenance System. Artificial Intelligence 24 (1986).

[Falkenhainer 86] B. Falkenhainer, K. D. Forbus, D. Gentner. The Structure-Mapping Engine. In *Proc. Fifth National Conference on Artifical Intelligence*, 1986.

[Fujita *et al.* 1991] H. Fujita, et. al. A Model Generation Therem Prover in KL1 Using a Ramified-Stack Algorithm. ICOT TR-606 1991.

[福井 1989] 福井. パーチャルタイムアルゴリズムの改良. 情報処理学会論文誌, Vol.30, No.12 (1989), pp. 1547-1554.

[Furuichi *el al.* 1990] M. Furuichi, K. Taki, and N. Ichiyoshi. A multi-level load balancing scheme for or-parallel exhaustive search programs on the Multi-PSI. In *Proc. of PPoPP'90*, 1990, pp. 50–59.

[Goto *et al.* 1988] Atsuhiro Goto *et al.* Overview of the Parallel Inference Machine Architecture. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1988*, ICOT, Tokyo, 1988.

[Hasegawa *et al.* 1992] Hasegawa, R. *et al.* MGTP: A Parallel Theorem Prover Based on Lazy Model Generation. To appear in *Proc. CADE' (System Abstract)*, 1992.

[広沢 1991] 広沢、星田、石川、戸谷：“３次元ダイナミックプログラミングを活用した蛋白質のアライメントシステム”，公開ワークショップ「ヒトゲノム計画と情報解析技術」論文集, 1991.

[Hirosawa *et al.* 1992] Hirosawa, H., Feldmann, R.J., Rawn D., Ishikawa, M., Hoshida, M. and Micheals, G. Folding simulation using Temperature parallel Simulated Annealing. In *Proc. Int. Conf. on Fifth Generation Computer System 1992*, ICOT, Tokyo, 1992.

[Ichiyoshi 1989] N. Ichiyoshi. Parallel logic programming on the Multi-PSI. ICOT TR-487, 1989. (Presented at the Italian-Swedish-Japanese Workshop '90).

[Ichiyoshi *et al.* 1992] N. Ichiyoshi and K. Kimura. Asymptotic load balance of distributed hash tables. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1992*, 1992.

[石川 1991] 石川、星田、広沢、戸谷、鬼塚、新田、金久：“並列推論マシンを用いたタンパク質の配列解析”, 情報処理学会 情報学基礎研究会報告 23-2, 1991.

[Jefferson 1985] D. R. Jefferson. Virtual Time. *ACM Transactions on Programming Languages and Systems*, Vol.7, No.3 (1985), pp. 404-425.

[Kimura *et al.* 1991] K. Kimura and K. Taki. Time-homogeneous Parallel Annealing Algorithm. In *Proc. IMACS'91*, 1991. pp. 827-828.

[Kimura *et al.* 1991] K. Kimura and N. Ichiyoshi. Probabilistic analysis of the optimal efficiency of the multi-level dynamic load balancing scheme. In *Proc. Sixth Distributed Memory Computing Conference*, 1991, pp. 145–152.

[Kitazawa 1985] H. Kitazawa. A Line Search Algorithm with High Wireability For Custom VLSI Design, In *Proc. ISCAS'85*, 1985. pp.1035–1038.

[Koseki *et al.* 1990] Koseki, Y., Nakakuki, Y., and Tanaka, M., An adaptive model-Based diagnostic system, In *Proc. PRICAI'90,* Vol. 1 (1990), pp. 104-109.

[Kumar *et al.* 1988] V. Kumar, K. Ramesh, and V. N. Rao. Parallel best-first search of state space graphs: A summary of results. In *Proc. AAAI-88*, 1988, pp. 122–127.

[Maruyama 1988] F. Maruyama et al. co-LODEX: a cooperative expert system for logic design. In *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, 1988, pp.1299-1306.

[Maruyama 1990] F. Maruyama et al. Logic Design System with Evaluation-Redesign Mechanism. Electronics and Communications in Japan, Part III: Fundamental Electronic Science, Vol. 73, No.5, Scripta Technica, Inc. (1990).

[Maruyama 1991] F. Maruyama et al. Solving Combinatorial Constraint Satisfaction and Optimization

Problems Using Sufficient Conditions for Constraint Violation. In Proc. the Fourth Int. Symposium on Artificial Intelligence, 1991.

[Matsumoto 1987] Y. Matsumoto. A parallel parsing system for natural language analysis. In *Proc. Third International Conference on Logic Programming*, Lecturel Notes on Computer Science 225, Springer-Verlag, 1987, pp. 396–409.

[Matsumoto *et al.* 1992] Y. Matsumoto and K. Taki. Parallel logic Simulator based on Time Warp and its Evaluation. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1992*, ICOT, Tokyo, 1992.

[Minoda 1992] Y. Minoda et al. A Cooperative Logic Design Expert System on a Multiprocessor. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1992*,ICOT,Tokyo, 1992.

[Nakakuki *et al.* 1990] Nakakuki, Y., Koseki, Y., and Tanaka, M., Inductive learning in probabilistic domain, In *Proc. AAAI-90,* Vol. 2 (1990), pp. 809-814.

[Needleman *et al.* 1970] Needleman,S.B. and Wunsch,C.D. A General Method Applicable to the Search for Similarities in the Amino Acid Sequences of Two Proteins. *J. of Mol. Biol.*, 48 (1970), pp. 443-453.

[Nitta *et al.* 1992] K. Nitta et. al. HELIC-II: A Legal Reasoning System on the Parallel Inference Machine. In *Proc. Int. Conf. on Fifth Generation Computer Systems 1992*, ICOT, Tokyo, 1992.

[沖他か 1989] 沖 廣明, 瀧 和男, 清 慎一, 古市昌一. マルチ PSI における並列詰め碁プログラムの実現と評価. 並列処理シンポジウム JSPP'89 論文集, pp. 351–357, 1989.

[Skolnick and Kolinsky 1991] Skolnick, J. and Kolinski,A., Dynamic Monte Carlo Simulation of a New Lattice Model of Globular Protein Folding, Structure and Dynamics, *Journal of Molecular Biology, Vol.221, No2*, pp.499-531.

[寿崎他か 1989] 寿崎かすみ, 佐藤裕幸, 杉村領一, 赤坂宏二, 瀧 和男, 山崎重一郎, 広田直人. マルチ PSI における並列構文解析プログラム PAX の実現と評価. 並列処理シンポジウム JSPP'89 論文集, pp. 343–350, 1989.

[Uchida *et al.* 1988] Shunichi Uchida et al. Research and Development of the Parallel Inference System in the Intermediate Stage of the FGCS Project. In *Proc. Int. Conf. on Fifth Generation Computer Systems*, ICOT, Tokyo, 1988.

[Ueda *et al.* 1978] Ueda, Y., Taketomi, H. and Go, N. (1978) Studies on protein folding, unfolding and fluctuations by computer simulation. A three dimensional lattice model of lysozyme. *Bilpolymers Vol.17* pp.1531-1548.

[Wada and Ichiyoshi 1990] K. Wada and N. Ichiyoshi. A study of mapping of locally message exchanging algorithms on a loosely-coupled multiprocessor. ICOT TR-587, 1990.

[和田他か 1992] 和田正寛, 六沢一昭, 市吉伸行. Iterative-Deepening A* アルゴリズムの並列化と並列推論マシン PIM/m 上の性能評価. 並列処理シンポジウム JSPP'92 論文集, 1992.