

演算時間及び通信量評価を可能とする並列計算機シミュレータの構築

村田 英明 浜村 秀彦
三菱重工業(株) 技術本部システム技術部

並列計算機の比較性能評価ツールとして、並列計算機シミュレータを構築した。本シミュレータは、並列プログラム及び並列アーキテクチャをデータとして入力されると、演算時間及び通信量を考慮したシミュレーションを行ない、並列計算結果及び予測処理時間を出力する。本シミュレータの有効性を評価するため、確率論的破壊力学計算及び連立一次方程式計算の並列処理開発への適用を行なった。その結果、本シミュレータは並列計算機の開発あるいは導入前に、並列アルゴリズム及び並列アーキテクチャの検証及び性能予測を可能とすることを示した。

A simulator evaluating the load of processors and communications between processors

Hideaki MURATA Hidehiko HAMAMURA
Mitsubishi Heavy Industries,LTD
1-1-1 Wadasaki-cho, Hyogo-ku, Kobe 652, Japan

To help the programmer who designs distribution algorithms on parallel computers, we have developed the simulator, which can calculate the total execution time of parallel programs by collecting the load of each processors and communications between processors. Simulations with various programs and architectures have indicated that the simulator is helpful to evaluate the performance of parallel processing and determine the fastest distribution algorithm and architecture.

1 はじめに

計算機システムの高速化技術の一つに並列計算機がある。画像処理やデータベースシステム等の特定の分野では、既に計算機メーカー各社が実用化システムを提供しつつあるが、ユーザが現在抱えている大規模な処理の全てを高速化する汎用並列計算機は、未だ研究段階にある。

並列計算機を適用し高速化を図る場合、

- 既に実用化された並列計算機の中で、高速化の対象となる問題に適するものを導入する。
- 適するものがない場合には、対象問題に適した並列計算機を自ら開発する。

の2つの方法が考えられる。

前者の方法では、実機導入前に種々の並列アルゴリズムと並列アーキテクチャを比較し、対象問題を高速に処理するような組合せを決定する必要がある。

しかしながら、並列計算機メーカーの提供するスペックから対象問題に対する性能を予測することは難しい。また複数の並列計算機を導入し、実際に並列処理を行なって比較することは非現実的である。

後者の方法においても、対象問題に適した並列アルゴリズム及び並列アーキテクチャの事前検証及び性能予測を十分に行なった上で、効率良く開発を行なうことが重要である。

当社では、米国 AT&T 社製の並列計算機ピクセルマシン [1] を導入し、これまでに画像処理 [2] や破壊力学計算等の高速化を行なってきた。しかし開発ツールの不備等により、高速な並列アルゴリズムの開発、特にデバッグやチューニングは容易ではなかった。また、ピクセルマシンで高速化できない問題に対しては、別の並列計算機の導入または開発が必要となる。

このような場合には、事前に性能予測を行なうこととなるが、その有効な手段の一つとしてシミュレータがある。製品化されている並列計算機には、開発ツールとしてシミュレータが付属することが多い。しかしながら、これらのシミュレータは特定の並列計算機を対象としており、他の並列アーキテクチャの性能予測に用いることはできない。

そこで、並列計算機の導入または自主開発を行なうユーザの立場から、並列アルゴリズム及び並列アーキテクチャの事前検証及び性能予測を目的として、並列計算機シミュレータの構築を行なった。

本稿では、並列計算機シミュレータの概要について述べ、適用例によってその有効性を示す。

2 並列計算機シミュレータ

2.1 概要

本シミュレータは EWS 上に仮想並列計算機として動作する。シミュレータの概要を図 1 に示す。

入力として、並列プログラム、並列アーキテクチャデータ、タスク割り当てデータを与える。

シミュレータは演算時間及び通信量を考慮しながら、並列プログラムを実行し、並列アーキテクチャのプロセッサ間通信のシミュレーションを行う。

出力として、並列プログラムの実行結果とともに、プロセッサやプロセッサ間通信路の実行状態の履歴や統計を得ることができる。これらの出力を用いて、並列アルゴリズム及び並列アーキテクチャの検証及び性能予測が可能となる。

2.2 内部動作

シミュレータ本体は、OS の提供する疑似並列処理環境上で、協調して動作する複数のプロセスである。

各プロセッサに割り当てられたタスクは、それぞれ 1 つの演算実行プロセスとして EWS の CPU で直接実行される。

演算実行プロセスで発生するプロセッサ間通信要求は、OS の提供するプロセス間通信機能によって、全て通信実行プロセスに、通信要求として送られる。

通信実行プロセスは通信要求を受け取り、並列アーキテクチャデータに従ってデータを転送する。

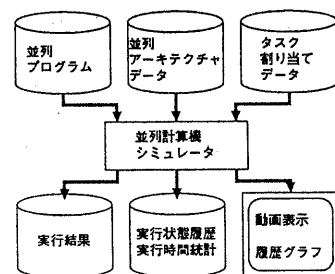


図 1: シミュレータの概要

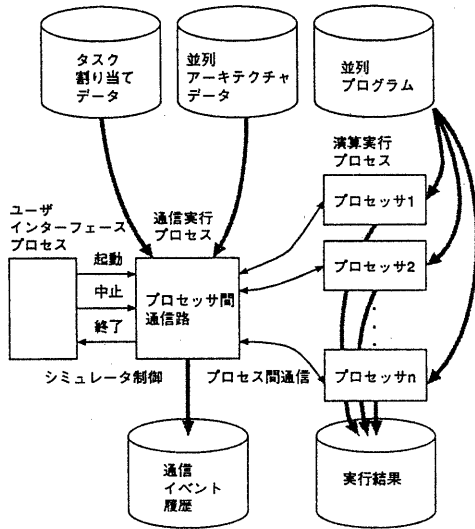


図 2: シミュレータの内部構成

即ち、演算実行プロセスは並列計算機の個々のプロセッサに対応し、プロセス間通信及び通信実行プロセスはプロセッサ間通信に対応することで、並列計算機全体のシミュレーションを実現する。シミュレータの内部構成を図 2 に示す。

2.3 性能予測手法

図 3 はタスク A とタスク B で構成される並列プログラムをプロセッサ A 及びプロセッサ B からなる並列計算機で実行する例を示している。

図 3(a) に示すように、タスク B は処理の途中でタスク A の処理結果を受信した後、処理を継続する。

図 3(c) の実機では、各プロセッサでの演算時間である T_{b1} 及び T_{a2} と、通信時間 T_c との和が全実行時間となる。

これに対し、図 3(b) に示すように、シミュレーションにおいても、シミュレータ上での演算時間及び通信時間を計測することができる。演算時間 T'_{b1} 及び T'_{a2} は演算実行プロセスでの計測、また通信時間 T'_c は通信実行プロセスでの計測である。これらの値に、シミュレータでの計測時間を実時間へ変換する関数を用いることにより、全実行時間の予測値を得ることができる。

2.4 入出力データ

2.4.1 並列プログラム

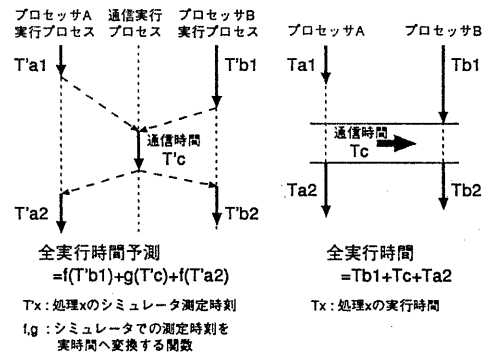
シミュレータの並列プログラムは、仮想的なポートへのデータの読み出しという形式でプロセッサ間通信を記述する。並列プログラムの記述例を図 4 に示す。

図 4 の例では、行番号 6 の部分において、port よりデータを受信する。各プロセッサ毎に、このような通信を行ないながら並列実行されるプログラムを記述する。

```
taskA()
{
    /* do a1 */
    CHwrite();
    /* do a2 */
}

taskB()
{
    /* do b1 */
    CHread();
    /* do b2 */
}
```

(a) 並列プログラム



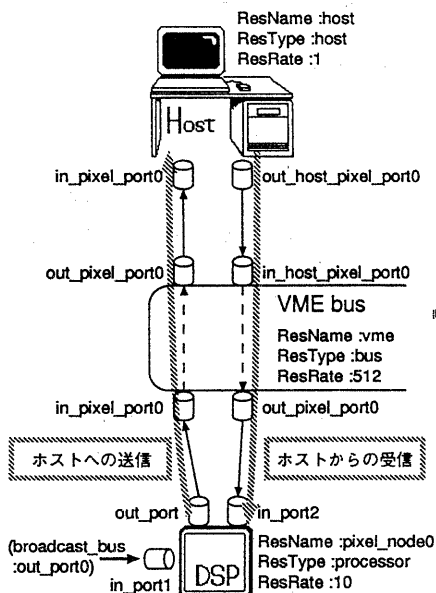
(b) シミュレータ

(c) 実機

図 3: シミュレータによる性能予測

```
1: int read_data(int port)
2: {
3:     char kf_databuf[4];
4:     int *kf = (int *)kf_databuf;
5:
6:     CHread(port, kf_databuf, 4);
7:     ...
8:     chkppm_(xx, itrial, kx, kf, rn, year);
9: }
```

図 4: 並列プログラムの記述例



```
# ホスト計算機の定義
#
ResName :host
ResType :host
ResRate :1

ExtConnect :in_pixel_port0:vme:out_host_pixel_port0
ExtConnect :out_host_pixel_port0:vme:in_host_pixel_port0
...

# バスの定義
#
ResName :vme
ResType :bus
ResRate :512
ExtConnect :in_pixel_port0:pixel_node0:out_port
ExtConnect :out_pixel_port0:pixel_node0:in_port2
ExtConnect :in_host_pixel_port0:host:out_pixel_port0
ExtConnect :out_host_pixel_port0:host:in_pixel_port0
IntConnect :in_pixel_port0:out_host_pixel_port0
IntConnect :in_host_pixel_port0:out_pixel_port0
...

# ピクセルノード0番の定義
#
ResName :pixel_node0
ResType :processor
ResRate :10
ExtConnect :in_port1:broadcast_bus:out_port0
ExtConnect :out_port:vme:in_pixel_port0
ExtConnect :in_port2:vme:out_pixel_port0
...
```

図 5: ピクセルマシンアーキテクチャの記述例

2.4.2 並列アーキテクチャ

本シミュレータに入力される並列計算機のアーキテクチャは、全てリソースの定義によって記述される。リソースとは、プロセッサ及び通信路をモデル化した、並列アーキテクチャの構成単位である。

各リソースでは、固有の処理速度が定義され、リソース種別がプロセッサあるいはホストの場合には演算速度を表すパラメータとなり、通信リソースの場合には、データ転送速度となる。また他のリソースとの接続関係が定義され、これらの情報を用いて並列計算機をシミュレートする。図 5 に並列アーキテクチャのリソースによる記述例を示す。

これ以外にも、多様な並列アーキテクチャを記述することができる。また、各リソースのパラメータあるいは接続関係を変更しシミュレーションを行うことで、並列アーキテクチャの設計変更の影響を予測することが可能となる。

2.4.3 タスクの割り当て

本シミュレータでは、プロセッサとプログラムの対応関係を記述したデータをシミュレータに入力する。図 6 にタスク割り当てデータの記述例をそれぞれ示す。

シミュレータはシミュレーション開始時に図 6 に示すようなデータファイルを読み込み、各行の左側に指定されたプロセッサ名に対応する演算実行プロセスとして、右側に指定されたプログラムを起動する。

```
# タスク割り当てデータ
#(プロセッサ名):(プログラム名)
pipe_node0 : pccf/plnode
...
pipe_node8 : pccf/plnode
pixel_node0 : pccf/pxnode
...
pixel_node64: pccf/pxnode
```

図 6: タスク割り当てデータの例

2.4.4 並列プログラム実行結果

並列計算機の各プロセッサで実行されるプログラムは、シミュレータ上においても直接実行され、その結果をファイルあるいはコンソールに出力する。

即ち、シミュレータは実機と同一の結果を得ることができ、並列アルゴリズム及び並列アーキテクチャの検証を行なうことができる。

2.4.5 通信イベント履歴

シミュレータは、発生したプロセッサ間通信を全てイベントとして記録する。通信イベントの内容には、発生リソース、通信元及び通信先、発生時刻等が含まれる。

この通信イベント記録を利用して、個々のプロセッサの演算及び通信時間の統計処理や並列処理状態の視覚化を行なう。

3 適用例

本章では、並列計算機シミュレータの適用例として、

- 確率論的破壊力学計算における並列アルゴリズムの性能予測
- 連立一次方程式計算における並列アーキテクチャの性能予測

について述べ、本シミュレータが並列アルゴリズム及び並列アーキテクチャの事前検証及び性能予測を行ない得ることを示す。

いずれの場合も、以下に示す手順でシミュレーションを行なった。

1. 並列アーキテクチャの定義データ及び並列プログラムを作成する。
2. 並列計算機シミュレータ上で並列処理を行なう。
3. 実行結果を逐次処理のそれと比較し、並列アルゴリズムの検証を行なう。
4. 通信イベント履歴より、総処理時間、各プロセッサでの実行時間統計を得て、並列処理の性能予測を行なう。

3.1 並列アルゴリズムの性能予測例

並列計算機ピクセルマシン上での確率論的破壊力学計算の並列処理を対象に、2つの並列アルゴリズムについての性能予測を行なう。

3.1.1 ピクセルマシンの並列アーキテクチャ

ピクセルマシンは82台のプロセッサを備えており、その内18台はパイプノードと呼ばれパイプライン接続されている。残りの64台はピクセルノードと呼ばれ格子状に接続されている。

ピクセルノードは、個々にホスト計算機とのデータの送受信を行うことができる。ピクセルマシンと外部との入出力は主にホスト計算機を介して行われる。ピクセルマシンの構成を図7に示す。

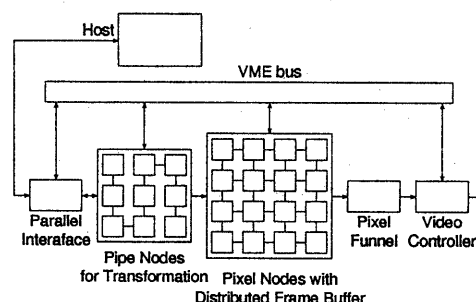


図 7: ピクセルマシンの並列アーキテクチャ

3.1.2 PCCF の並列アルゴリズム

確率論的破壊力学コード PCCF([3]) は、半楕円亀裂付き平板モデル (図 8 参照) において引っ張り荷重及び曲げ荷重が繰り返し負荷されたときのクリープ疲労亀裂進展による破損確率をモンテカルロ法を用いて、評価するコードである。

PCCF では非常に多くの確率的試行 (通常 10^4 程度) を繰り返すことにより解析計算を行うため、その計算量は膨大である。

しかしながら、各試行は互いに独立であり処理依存関係は存在しない。従って繰り返し処理単位毎に分割することによって得られるタスクは、並列処理が可能となる。

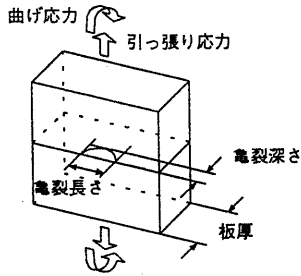


図 8: 半楕円亀裂付き平板モデル

ここで、PCCF の並列アルゴリズムにおいて、ホスト計算機が各プロセッサより解析計算の結果を受信した後、新たな確率変数を各プロセッサへ送信する場合に、次の 2 つの方式がある。

- (方式 1): 全プロセッサより結果を受信した後、新たな確率変数を各プロセッサへ順次送信する。
- (方式 2): 各プロセッサに対し結果を受信次第、直ちに新たな確率変数を送信する。

各方式についてプロセッサの状態変化を模式的に表したものを図 9 に示す。

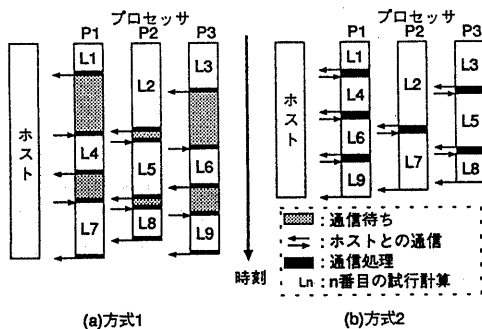


図 9: 送信方式の比較

定性的には、方式 2 の方が通信によるオーバーヘッドが少なく、処理速度の点で有利であることを推察できる。しかしながら、ピクセルマシンでは演算時間及び通信時間を容易に計測することができないため、並列計算機シミュレータによる性能予測が有効となる。

3.1.3 シミュレーション結果

ピクセルマシンの並列アーキテクチャ定義データと 2 種類の並列プログラムをシミュレータに与え、並列処理のシミュレーションを行なった。

(1) 総処理時間

370 回の試行計算を行なった場合の処理時間を表 1 に示す。

また、括弧 () 内の数値は実測に基づく実時間への換算時間である。PCCF コードについては、既に実機での並列処理を行なっている。実機での総処理時間、演算時間、通信時間の計測値とシミュレータでの計測値との比較により、 $t_u = 2.9 \times 10^{-4} [\text{sec}/\text{時刻}]$ を得た。これを性能予測の際の換算値として用いた。

ピクセルノード 32 台及び 64 台のいずれの場合も、方式 2 の方が処理速度が向上していることが分かる。

表 1: 処理時間の比較

ピクセル ノード数	方式 1 [時刻] ([sec])	方式 2 [時刻] ([sec])
32	5834 (1.69)	3862 (1.12)
64	3459 (1.00)	2388 (0.69)

(2) プロセッサ個別の実行時間統計

並列計算機シミュレータによって総実行時間、通信待ち時間、通信時間の統計データの一部として、ピクセルノード 0 番を表 2 に示す。表 2 のデータは、方式 1 が通信待ち時間が大きいアルゴリズムであることを定量的に示す。

表 2: 通信時間の比較 (ピクセルノード 0 番)

各状態時間	方式 1[時刻]	方式 2[時刻]
解析計算実行	3020	3571
通信待ち時間	2729	89
通信時間	27	36
総実行時間	5796	3725

このように本シミュレータは、並列アルゴリズムの全体の処理時間の性能予測を可能にするとともに、個々のプロセッサについてもデータを収集することができる。

3.2 並列アーキテクチャの性能予測

連立一次方程式計算の並列処理を対象に2つの並列アーキテクチャの性能予測を行なう。

3.2.1 ガウス消去法の並列アルゴリズム

図10に示すガウス消去法では、行列よりピボット行と呼ぶ着目行を選び出し、ピボット以外の行との加減乗除演算によって項を消去し、解を求める。従って、ピボット行を選択した後の各行の消去操作は、互いに並列に実行可能であることがわかる。以下にガウス消去法の並列アルゴリズムの概略を示す。

1. 係数行列を行方向に分割し、各プロセッサに割り当てる。
2. マスタプロセッサは、各プロセッサよりピボット候補を集め最大のピボット値を求めた後、そのピボット行データを全プロセッサに送信する。
3. 各プロセッサは、受信したデータを用いて割り当て行の消去操作を行なう。
4. 全ての行データの消去が終了するまで2~3を繰り返す。

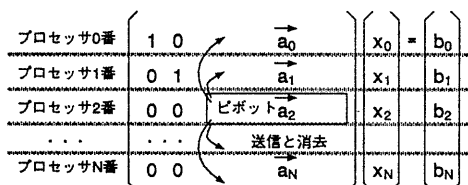


図 10: ガウス消去法

3.2.2 2つの並列アーキテクチャ

ガウス消去法の並列アルゴリズムでは、マスタプロセッサと他のプロセッサの間に頻繁な通信が発生する。従って、高速に通信できる並列アーキテクチャが要求される。

ここで、ピクセルマシン上でのガウス消去法の並列処理を考えると、

- マスタプロセッサから全プロセッサへの送信
→ ホスト計算機からパイプラインノード-ブロードキャストバス経由での全ピクセルノードへの送信
- 各プロセッサからマスタプロセッサへの通信
→ ピクセルノードから VME バス経由でのホスト計算機への送信

によって並列処理を実現できる。

しかしながら、ピクセルマシンではブロードキャストバスを利用するために、パイプラインノードを経由しなくてはならず、その通信時間が並列処理のオーバヘッドとなる。

また、各ピクセルノードがブロードキャストバスを利用するためには、ホスト計算機を介する必要がある、やはり並列処理による高速化の妨げとなる。

そこで、ピクセルマシンの並列アーキテクチャから、パイプラインノードを取り去り、ホスト計算機とブロードキャストバスを直接接続するとともに、各ピクセルノードがブロードキャストバスに直接データを出力可能なアーキテクチャを新たに提案し、図11(b)に示す。

以後、図11(b)のアーキテクチャをバス直結型と呼び、ピクセルマシンと同様の図11(a)のアーキテクチャをピクセルマシン型と呼ぶこととする。

3.2.3 シミュレーション結果

2種類の並列アーキテクチャ定義データと、ガウス消去法の並列プログラムをシミュレータに与え、並列処理のシミュレーションを行なった。

(1) 総処理時間

プロセッサ16台で、100元の連立一次方程式の計算を行なった場合の処理時間を表3に示す。破壊力学計算への適用例の場合と同様に、 $t_u = 2.9 \times 10^{-4} [\text{sec/時刻}]$ を用いた換算時間を併せて示した。

(2) プロセッサ個別の実行時間統計

表 3: 処理時間の比較

	ピクセルマシン型	バス直結型
処理時間	25066[時刻]	13617[時刻]
換算時間	7.3[秒]	3.9[秒]

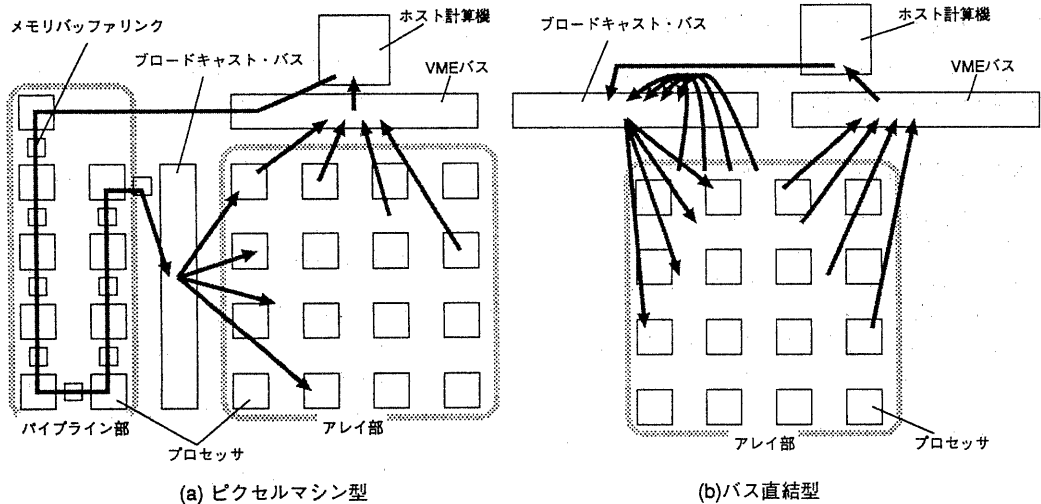


図 11: ピクセルマシン型とバス直結型

総実行時間、通信待ち時間、通信時間の統計データの一部として、プロセッサ 0 番を表 4 に示す。

ピクセルマシン型では、パイプラインを介した通信による通信待ち時間が大きいため、バス直結型に比べて処理効率の低いアーキテクチャであることが分かる。

表 4: 通信時間の比較 (プロセッサ 0 番)

各状態時間	ピクセルマシン型 [時刻]	バス直結型 [時刻]
解析計算時間	1200	1360
通信待ち時間	23252	11694
通信時間	521	561
総実行時間	24973	13615

このように、本シミュレータは並列アーキテクチャの定量的な性能予測を可能にするため、高速な並列アーキテクチャの設計に有効である。

4 おわりに

並列アルゴリズム及び並列アーキテクチャの事前検証及び性能予測を実現する並列計算機シミュレータの構築について述べ、その有効性を適用例によって示

した。

今後は、既に製品化あるいは発表されている他の並列アーキテクチャを積極的にデータ化し、本シミュレータを用いて評価することで、その幅広い有用性を確認したい。

また、本シミュレータの機能を拡張し、並列アルゴリズム及び並列アーキテクチャの性能のボトルネックの自動解析ツールへと発展させたい。

参考文献

- [1] Pixel Machines DEVtool User's Guide, AT&T
- [2] 古結, 浜村, “Na 中透視装置 (USV) における開口合成の並列処理”, 電子情報通信学会技術研究報告, CPSY91-21(1991-7)
- [3] G. Yagawa et al., “Applications of Probabilistic Fracture Mechanics to FBR components”, SMiLT91, Trans. vol. G, G-30(M)2, pp.337-342, 1991