ハイパフォーマンス
コンピューティング 77−31
（1999. 8. 4）

分散処理環境における数値シミュレーションの静的負荷分散手法

山下真史† 市川周一†

処理能力の異なる複数のプロセッサ (PE) からなる分散処理環境上で，通信時間と計算時間の双方を考慮して並列数値シミュレーションを静的に負荷分散する手法について述べる．この問題を，各計算ブロックへのPEの分配と，PEの処理能力に合わせた計算ブロックの分割，という2段階に分けて解決する．いずれの問題も計算が極めて困難なので，分枝限定法や近似アルゴリズムを利用して現実的な時間内で解を求める．シミュレーションによれば，ブロック数8，プロセッサ数24という条件下で提案する近似アルゴリズムの誤差は最適解から約5%であった．近似解の求解時間は現状の計算機でも1秒未満と充分実用的である．

Static load-balancing for distributed processing of numerical simulations

SHINJI YAMASHITA and SHUICHI ICHIKAWA†

This paper describes a static load-balancing scheme for parallel numerical simulations on distributed computing environment, which usually has a variety of processing elements (PEs). This problem is solved by the following two steps: (1) PEs are distributed among computing blocks, (2) then each computing block is split for each PE to minimize processing time of the whole simulation, considering both of computation and communication. As this problem is a kind of combinatorial optimization which is very hard to solve, this paper also shows some algorithms which gives good approximation in reasonable time.

1. はじめに

市川ら[1]は，偏微分方程式 (PDE) の並列求解システム NSL を例にとって通信時間と計算時間の双方を考慮する静的負荷分散法を示した．NSL[2]ではPDEを差分式に変換して陽解法で解く．計算対象となる物理領域は，境界適合法とマルチブロック法によって，相互に接続された複数の矩形の計算領域(ブロック)に写像される(図1)．

各ブロックは2次元に配列された格子点からなり，各格子点上ではPDEに対応した差分式が計算される．差分式の計算には近隣の格子点のデータが必要であるが，NSLで採用している陽の差分式では時刻 t における各格子点の計算に相互のデータ依存性がなく，全て並列に実行することができる．従って，各ブロックを複数の要素プロセッサ (PE) に割り当てることで差分式の計算を並列に実行できる．ただし，各時間ステップの計算後には格子点の値を交換するためにPE間通信が発生する．論文1)は，このような並列計算モデルに関して実行時間を最小にする静的負荷分散法を論じている．

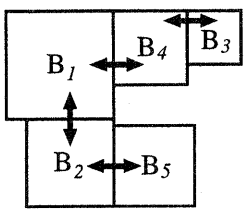

図1 計算領域(ブロック)

論文1)では並列計算機の利用を前提としているので，PEの構成・性能が全て等しいと仮定していた．しかし近年注目されているPCクラスタなどの分散処理環境では，一般にこの仮定が成り立たない．本研究では，PEの性能が不均一な分散処理環境に論文1)の静的負荷分散法を拡張する方法について論ずる．

2. 計算モデル

本研究で扱う計算モデルは論文1)の計算手順1を，分散環境に適応するよう拡張したものである．本章のモデルと論文1)のモデルの最も大きな相違は，並列計算機のPEは区別の必要がないのに対し，分散環境ではPEの能力が不均一なので全てのプロセッサを区別する必要があるという点である．

† 豊橋技術科学大学 工学研究科 知識情報工学専攻
Department of Knowledge-based Information Engineering, Toyohashi University of Technology

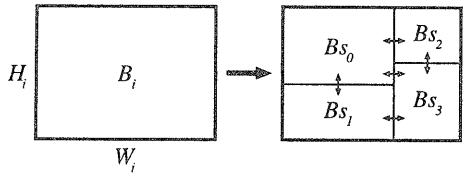

図2　ブロックの分割

2.1　負荷分散法の概要

以下，ブロック数を m，PE数を n として，計算負荷を分散する方法を考える．なお本研究では論文1) と同じく $m \leq n$ と仮定し，各ブロックに対して1つ以上のPEを割り当てる．ブロックを担当するPEが複数ある場合，ブロックをPEの数に分割して，それぞれの断片(サブブロック)をPEに割り当てることにする (図2)．

分割にあたってはサブブロックは必ず矩形にするとする．自由な形状に分割することを許すと，通信時間が増大したり並列処理ソフトウェアが複雑化する可能性がある．逆に矩形であればベクトル処理等による高速化も期待できる．同様な理由により，複数のブロックから同一のPEにサブブロックを割り当てることを禁じる．この制限によって負荷分散の質は若干損なわれるが，並列処理ソフトウェアの構造は単純になり，また通信時間や計算時間のモデル化が容易になるため負荷分散や最適化が精度良く行えると考えられる．このような制限のない一般的な負荷分散に関しては，現在別に研究中である[3]．

本研究の目的である負荷分散問題を解くためには，まず部分問題として，「ブロックを担当するPE(1つ以上)が決まったとき通信時間と計算時間の両方を考慮して処理時間が最小になるようにブロックを分割する方法」を求めなければならない．この部分問題については3章で検討する．この部分問題が解決すれば，あとは「n 個の区別のあるプロセッサを m 個の区別のあるグループに分割する方法のうちから実行時間を最短にする組合せを選ぶ」という組合せ最適化問題に帰着される．この組合せ最適化問題は $O(n^m m^{n-m})$ の探索空間を持つと考えられ，極めて計算が困難な問題である．4章では，この最適化問題が分枝限定法[4)5)] の採用により実用的な時間で解けることを示す．また，精度が良く実行時間の短い近似アルゴリズムに関しても説明する．

2.2　評価関数について

本研究の目的は，分散処理環境で数値シミュレーションの実行時間を最小化することである．したがって，本研究では実行時間 T を評価関数とする．既に述べたように，本論文の計算モデルでは各PEがサブブロック1つを担当するので，対応するPEとサブブロックに同じ添字 i を付けて区別する．最も実行時間の長いサブブロックがクリティカルパスとなるため，PE数を n とすれば全体の実行時間 T は以下の式で表される．T_i は PE_i の実行時間である．

$$T = \max_i T_i \qquad (i = 0, 1, \ldots, n-1) \tag{1}$$

$$T_i = Ta_i + Tc_i \tag{2}$$

計算時間 Ta_i はサブブロック Bs_i の格子点数 Sa_i の一次関数であるとする．

$$Ta_i = Cta_i Sa_i + Dta_i \tag{3}$$

ここで Cta_i は格子点あたりの計算時間である．PE_i の処理速度が個別に違うので，Cta_i もPE毎に異なる．Dta_i はサブブロックの計算による遅延を表す定数項である．

通信時間 Tc_i は通信量 Sc_i の一次関数と見積もれるので，

$$Tc_i = CtcSc_i + Cn_i Dtc \tag{4}$$

となる．本論文では問題を単純化するために，小規模PCクラスタなど均一なネットワークを持つ環境を仮定することとする．従って帯域幅 Ctc，通信による遅延 Dtc はすべてのPEで等しいと仮定した．Cn_i はサブブロック Bs_i のPE間通信の数で，サブブロックの配置によって異なる．例えば，図2の Bs_0 では $Cn_0 = 3$ となる．通信量 Sc_i は，サブブロックの縦横の格子点数をそれぞれ h_i, w_i とするとき以下の式で表される[1)]．

$$Sc_i = 2\delta\,(h_i + w_i + 2\delta) \tag{5}$$

3.　ブロック内の負荷分散

本章では，単一のブロックを処理能力の異なる複数のPEに割り当てるためのブロック分割方法について検討する．このブロック分割は，図形的制約のある一種の組合せ最適化である．例えば図2のように分割するとき，各サブブロックの辺長は整数でなければならない(整数制約)．しかも各サブブロックをパズルのように組み合わせて元のブロックが構成できなければならない(図形的制約)．ブロック全体の実行時間を最小化するには，通信と計算を考慮して，各PEの実行時間の最大のものが最小となるようなブロック分割を求めれば良い．

しかしこのような最適化問題を解くのは極めて困難である．そこで本研究では，ヒューリスティックを用いた近似的分割法を検討する．近似的分割法としては以下の2つをとりあげるが，いずれも切断面が直線であると通信量が減るという直観に従って，計算量のバランスを取りながら再帰的に直線分割していく手法である．これは一般にRB (Recursive Bisection)[6)7)] と呼ばれる方法の一種である．これらの近似アルゴリズムの結果と，緩和問題を解いて求めた下界値を比較して，近似的分割法の精度を評価する．ここでは紙数の都合上，基本的なアイディアだけを紹介する．

3.1　近似的分割法

3.1.1　分割法1 (type1)

PEを2つのグループに分け，グループ全体の処理速度に比例した格子点数が分配されるよう，ブロックを直線的に二分する．このときブロックの縦方向と横方向の

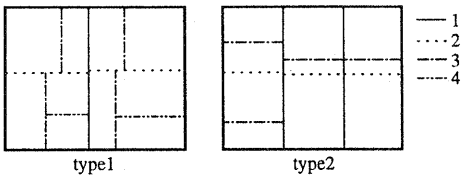

図3 近似的分割法の分割例

うち通信量の少ない方向に切る．これを1グループ1PEになるまで再帰的に繰り返す．分割例を図3のtype1に示す．1回目から4回目までの分割線を，線の種類で図中に示した．

3.1.2 分割法2 (type2)

PEがn個あるとき，まずPEを$\lfloor\sqrt{n}\rfloor$個のグループに分ける．各グループの処理能力に応じて，$\lfloor\sqrt{n}\rfloor-1$本の平行な直線でブロックを分割する．この時も縦横のうち通信量の少ない方向に切る．これ以後は，各グループに対して分割法1を適用する．分割例を図3のtype2に示す．分割法2は同じ能力のPEが多数あるとき効果があると考えられる．

3.1.3 局所的負荷調整

分割法1と分割法2では計算時間の均衡だけを考えて分割している．そのためPEの処理能力が大きいと，対応するサブブロックの格子点数も大きくなり，結果的に通信量も増加する．通信性能はPEの能力によらずネットワークで決まるので，計算時間が均等になるように分割すると速いPEほど通信時間が大きくなって合計処理時間が平均より大きくなってしまう．そこで処理時間を元に近似的分割を求めた後，大きなサブブロックから隣接するサブブロックに一部の格子点を移動して局所的負荷調整を行う (図4)．

(1) 各ブロックの中で実行時間が最大のサブブロックBs_iを求める
(2) Bs_iと隣接しているサブブロックの中で実行時間が最小のサブブロックBs_jを求める
(3) Bs_iとBs_jの実行時間が均等になるようにBs_iの格子点をBs_jに移動
(4) (1)から(3)を繰り返す
(5) 移動できる格子点がなくなったら終了

ただし，サブブロックの境界の形によって格子点を移動できるサブブロックが限られる．図2の場合は，Bs_0とBs_1，Bs_2とBs_3の間でのみ格子点の移動が可能である．

3.2 緩和問題

近似的分割法の評価基準として，緩和問題を解いて実行時間の下界値を求める．緩和問題は以下のように定義する．

- サブブロックの辺長は整数で無くても良い．
- サブブロックの面積の和が元のブロックと等しければ良い．

一番目の条件で整数条件を外し，二番目で図形的制約を外す．この緩和問題では格子点数は減らない (計算時間は減らない) ので，通信時間を最小にすると実行時間が最小になる．従って，各サブブロックを正方形にして計算量に対する通信量を最小にし，さらに各プロセッサでの計算時間と通信時間の和が最小になるように格子点数を割り当てれば良い．この問題は非線形計画問題であるが，整数制約がなく評価関数が凸なので，簡単に解くことができる．

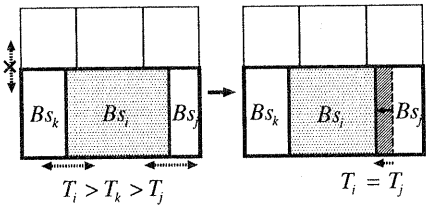

図4 局所的負荷調整

3.3 評価結果

図5と図6に，以上の近似的分割法の適用結果を示す．図中のtype1は分割法1，type2は分割法2，localは局所的負荷調整を行ったものを示す．結果は下界値を1として正規化してある．

図5では，全てのPEで$Cta_i = 0.01$とし，ブロックは$W = 500, H = 400$として，PEの台数を変えながら近似アルゴリズムの精度を検証した．分割法2は分割法1より精度が良いが，局所的負荷調整を適用してもそれ以上の改善がない．

図6では，PEの能力が不均一な場合について検討した．PE数は10で，Cta_iの内訳は表1の通りである．シミュレーションではブロックの縦横比を$W : H = 5 : 4$に保持したままブロックのサイズを変化させて近似解の精度を調べた．ここでは分割法1と2でほとんど差がない．しかしどちらも局所的負荷調整による改善が見られる．

これらの結果から，分割法2が分割法1より優れていると判断した．

4. ブロック間の負荷分散

本研究では各ブロックへの最適なPEグループの割り当てと最適なブロック分割を行って，実行時間を最小にすることを目的としているが，第3章で述べたように，最適なブロック分割を求めるのは困難である．そのため，第3章の近似的分割法により得られたブロック分割

表1 PEの内訳

Cta_i	台数
0.0100	3
0.0050	3
0.0033	2
0.0025	1
0.0020	1

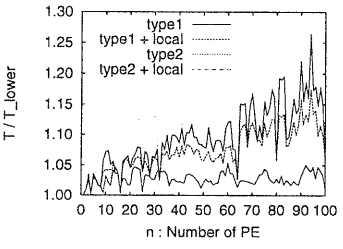

図5 各PEの能力が均一な場合

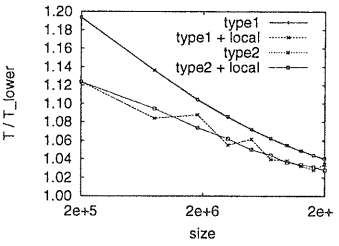

図6 PEの能力が不均一な場合

が最適なものと仮定してその上で実行時間を最小にすることを考える．

4.1 分枝限定法

前にも述べたように，この問題の探索空間は膨大であるため，すべての組合せを調べることは現実的でない．そこで分枝限定法[4)5)]を用いて探索空間を制限する．

暫定解を $\bar{T}$，最適解を τ，第3章で用いた下界値を T_{lower} とすると，

$$\bar{T} \geq \tau \geq T_{lower} \tag{6}$$

が成り立つ．既に3.2節で下界が求まっているので，T_{lower} にはその下界値を用いればよい．ある組合せに対して T_{lower} を計算し，その値が暫定解 $\bar{T}$ より大きいときは，その組合せで最適解が得られる可能性はない．また，目的関数 T は式(1)により求められるので，あるサブブロックの実行時間 T_i が暫定解より悪くなった時点で他のサブブロックの計算を打ち切る．もし $\bar{T}$ よりよい実行可能解を得られれば，$\bar{T}$ を更新する．こうして動的に探索範囲を狭めることができる．

4.2 プロセッサ分配の近似アルゴリズム

分枝限定法では，暫定解が悪いと探索空間が広くなり，暫定解の更新がますます難しくなる．そのため探索の初期から比較的最適解に近い暫定解を用いることが，探索時間を短縮する鍵となる．このため精度の良い近似アルゴリズムが必須となる．近似アルゴリズムでは短時間で実行可能な解を求めることが必須で，解の質がよく，精度保証があることが望ましい．

本研究ではヒューリスティックによる近似アルゴリズムを用いる．精度保証はないが，短時間で実行可能な解が求まる．この近似アルゴリズムと局所探索によって得られた実行可能解を分枝限定の暫定解の初期値とする．近似アルゴリズムの評価は第5章で行う．

4.2.1 近似アルゴリズム1

簡単な近似アルゴリズムとして次のようなものが考えられる．ブロックを格子点数で降順にソートする．PEは Cta_i で昇順にソートする(処理速度で降順ソート)．そして PE_i を B_j $(j = i \bmod m)$ に割り当てる(図7)．これを近似アルゴリズム1(approx1)とする．

4.2.2 近似アルゴリズム2

次に各ブロックの計算量と各PEの性能の不均一性を考慮し，計算量の多いブロックから順に速いPEを割り当てていく方法を考える．これも近似アルゴリズム1と同じように，ブロックとPEをソートする．ブロック B_i の格子点が全体の格子点に占める割合を RB_i，PE_i の処理速度が全体の処理速度(総和)に占める割合を RPE_i は，

$$RB_i = H_iW_i \Big/ \sum_{j=0}^{m-1} H_jW_j \tag{7}$$

$$RPE_i = \frac{1}{Cta_i} \Big/ \sum_{j=0}^{n-1} \frac{1}{Cta_j} \tag{8}$$

となる．図8の矩形の大きさはそれぞれその割合を示している．ブロックの割合 RB_i の大きい順に下の手順でPEを割り当てる(図8)．

(1) 初期値 $i = 0, j = 0$
(2) B_i に PE_j を割り当て，$RB_i := RB_i - RPE_j$ を計算
(3) まだPEが1つも割り当てられていないブロック数 $m - i - 1$ と，まだ割り当てられていないPEの数 $n - j - 1$ が等しくなったら，それらのブロックにPEを1つずつ順番に割り当てて終了
(4) $j := j + 1$
(5) $RB_i > 0$ なら(2)へ
$RB_i \leq 0$ なら(6)へ
(6) $i := i + 1$ して(2)へ
(7) 全PEをいずれかのブロックに割り当てたら終了

これで各ブロックに最低1個のPEが割り当てられる．これを近似アルゴリズム2(approx2)とする．

4.2.3 近似アルゴリズム3

また，近似アルゴリズム2に多少の変更を加えた

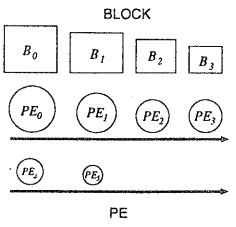

図7 近似アルゴリズム1

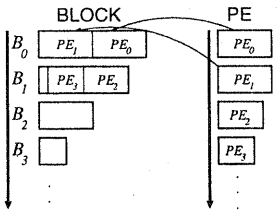

図8 近似アルゴリズム2

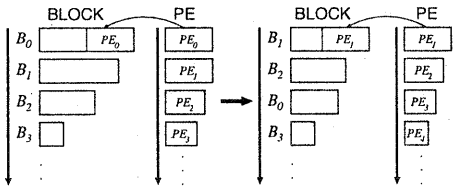

図9 近似アルゴリズム3

ものを考える．近似アルゴリズム2と同じように，RB_i, RPE_i を計算し，降順にソートする．そして以下の手順でサブブロックにPEを割り当てる(図9)．

(1) 初期値 $i=.0, j=0$

(2) B_i に PE_j を割り当て，$RB_i := RB_i - RPE_j$ を計算

(3) まだPEが1つも割り当てられていないブロックの数と，まだ割り当てられていないPEの数 $n-j-1$ が等しくなったら，それらのブロックにPEを1つずつ割り当てて終了

(4) RB が最大のサブブロック B_k を求め，$i := k$ とする

(5) $j := j+1$ して(2)へ

(6) 全PEをいずれかのブロックに割り当てたら終了

これで各ブロックに最低1個のPEが割り当てられる．これを近似アルゴリズム3(approx3)とする．

4.3 局所探索

これらの近似アルゴリズムにもブロック分割と同様に局所探索を行う．

(1) 近似アルゴリズムによりPEのグループ $P_i(i=0,1,\ldots,m-1)$ を得る

(2) 各ブロック B_i を P_i で分割$(i=0,1,\ldots,m-1)$

(3) 各サブブロックの実行時間$T_j(j=0,1,\ldots,n-1)$を計算

(4) その中から実行時間が最小のサブブロック Bs_a を持つブロック B_A と，最大のサブブロック Bs_b を持つブロック B_B を探す
$Bs_a \in B_A, Bs_b \in B_B \quad (A \neq B)$

(5) P_A の要素で Cta_i が一番大きなものを P_B に加える

(6) P_A の要素で Cta_i が一番大きなものと P_B の要素で Cta が一番小さなものを入れ替える

(7) (5)か(6)で T が改善されたら，良い方を選択して，P_A と P_B を更新し(2)へ

(8) T が改善されなくなったら終了

5. 評　価

5.1 評価方法

数値シミュレーションにより本研究で提案した手法を評価する．ブロック数 m，PE数 n，PEの処理速度 Cta_i とブロックサイズ H_i, W_i を変えながらシミュレーションを行い，分枝限定法で求めたブロック分割と，近似アルゴリズムで求めた分割結果の比較を行った．

ブロック数は $m=4,8$ の2通りとし，PE数 n は4の倍数として，4から4つずつ増やした．それ以外のパラメータは表2の通りである．PEの処理速度 Cta_i は表2に示す通り4種類とし，$Cta_0, Cta_1, Cta_2, Cta_3$ をそれぞれ $n/4$ 台ずつとした．また Dta_i はプロセッサによらず同一とした．これらのパラメータは実装依存であるが，本研究では特定の実装によらず一般的と思われる値を設定した．

ブロック B_i の縦方向の格子点数 H_i，横方向の格子点数 W_i(図2)は，以下の条件に合わせて乱数で生成した．最適化問題の解はデータ依存であるため，各(m,n)について20回の試行を行い平均値をとって評価する．

$$100 \leq H_i, W_i \leq 1000 \tag{9}$$

$$H_i, W_i \equiv 0 \pmod{10} \tag{10}$$

5.2 近似アルゴリズムの精度

第4.2節で述べた近似アルゴリズムを，分枝限定法で求めた最適解と比較して評価する．ただし各ブロック内での負荷分散には，第3章の分割法2による解を用いている．図10に $m=4$，図11に $m=8$ の近似アルゴリズム精度を示す．図中のlocalは各アルゴリズムに局所探索を行った結果である．結果は最適解を1として正規化してある．

図10$(m=4)$では n が m より大きくなるほど近似解の精度が良くなっていることがわかる．approx2,3とも最適解との誤差が約10%以内に収まっている．さらに局所探索により数%の改善が見られる．3つの局所探索により得られた近似解のうち，最も良いものをとったものが図中のbest effortである．best effortでは最適解との誤差が2,3%以内に収まっている．図11$(m=8)$では n が m に対してそれほど大きくないので近似解の精度が安定していない．しかし局所探索により，誤差が5%くらいまで改善されている．best effortでは $m=4$ の場合と同じく，2,3%くらいの誤差である．

図10と図11より，これら3つの近似アルゴリズムと

表2 パラメータ一覧

名前	値	名前	値
Cta_0	0.0050	Ctc	0.2
Cta_1	0.0033	Dtc	0.1
Cta_2	0.0025	Dta_i	10.0
Cta_3	0.0020	δ	1

局所探索によって得られた近似解は最適解から2,3%の誤差であり，十分な精度が得られていることがわかる．

5.3 負荷分散の所要時間

次に最適解の求解時間を図12に示す．実行時間はデータ依存なので，ブロックを乱数で生成し20回の試行の平均時間で評価する．図中のapprox.は3つの近似アルゴリズムの合計実行時間を示し，optimizationは最適解を求めるのに要した時間である(ただし近似アルゴリズムの実行時間は除く)．測定に使用した計算機環境はIntel Pentium2 400MHz，主記憶256MB，FreeBSD 3.1，GNU Cコンパイラ(ver 2.7)である．

近似アルゴリズムの実行時間，組合せ最適化の実行時間ともにnが増えると指数的に増大している．しかし組合せ最適化の実行時間の増大率は近似アルゴリズムを上回っており，nが増えるにつれ近似アルゴリズムとの差が大きくなっているのがわかる．近似アルゴリズムの実行時間は，最適化と比べると十分に実用的な時間である．

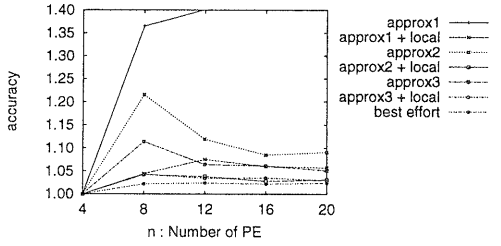

図10 PE数nと近似アルゴリズム精度(m=4)

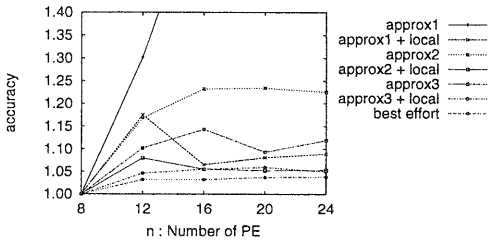

図11 PE数nと近似アルゴリズム精度(m=8)

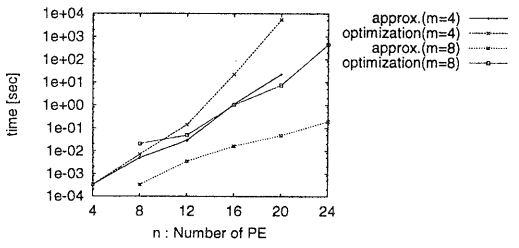

図12 シミュレーションの実行時間

以上の結果から，精度と実行時間の両面で本論文で提案する近似アルゴリズムの有用性が確認された．

6. おわりに

本研究では分散処理環境における数値シミュレーションの静的負荷分散法について検討した．分散処理環境は最適化の対象として考えたとき自由度が非常に大きく，また通信などのモデル化が非常に難しい．今後，どのようなモデルが現実的であるか，またどのような条件で評価すべきなのか，実測を含めてさらに検討してゆく必要がある．

謝辞 本研究の一部は，(財) 電気通信普及財団・平成10年度研究助成，文部省科学研究費補助金・特定領域研究 (B) (2) 10205210および奨励研究 (A) 11780211によるものである．

参考文献

1) 市川周一, 川合隆光, 島田俊夫: 組合せ最適化による並列数値シミュレーションの静的負荷分散, 情報処理学会論文誌, Vol. 39, No. 6, pp. 1746–1756 (1998).
2) 川合隆光, 市川周一, 島田俊夫: 並列数値シミュレーション用高水準言語 NSL, 情報処理学会論文誌, Vol. 38, No. 5, pp. 1058–1067 (1997).
3) 藤村佳克, 市川周一: 並列数値シミュレーションの静的負荷分散法の 拡張について, 並列処理シンポジウム JSPP'99, 情報処理学会, p. 203 (1999).
4) 茨木俊秀: 組合せ最適化, 産業図書 (1983).
5) 坂和正敏: 数理計画法の基礎, 森北出版, chapter 3.4, pp. 111–132 (1999).
6) Fox, G. C., Williams, R. D. and Messina, P. C.: *Parallel Computing Works!*, Morgan Kaufmann, chapter 11.1.5 (1994).
7) Fox, G. C.: A graphical approach to load balancing and sparse matrix vector multiplication on the hypercube, *Numerical Algorithms for Modern Parallel Computer Architectures* (Schultz, M.(ed.)), Springer-Verlag, pp. 37–62 (1988).