動的計画法の並列計算†

——並列計算性とアルゴリズム——

瀬口靖幸††* 田中正夫†† 中島利朗†††**
金田悠紀夫†† 小畑正貴††††*** 西野佐登史†††††***

動的計画法はすぐれた最適化手法の一つであるが，その計算時間は問題の規模とともに急速に増大する．そこで本研究では，近年注目をあつめている並列計算方式を採用することで，動的計画法計算の高速化をはかるための基礎として，その並列計算可能性とアルゴリズムについて考察した．まず標準的直列型問題の表解法について検討し，状態および決定変数についての繰返し計算が，完全に並列計算可能であり，段についての繰返し計算も部分利得関数の同期をとることで並列性を得ることが可能であることがわかった．また，この問題の多段決定過程への分解過程に注目し，小規模な動的計画問題を並列的に処理した後に，原問題の解を求めうることから，表解法とは異なる視点からの並列計算可能性を示した．さらに，非直列型問題を反復的に解く発見的手法の一つについて考察し，状態変換過程を制限する方法と組合せることで，高い並列計算性を有する構造が得られ，そのアルゴリズムを示すことができた．提案した表解法ならびに反復的解法の並列アルゴリズムを簡単な配分問題および一次方程式の数値計算に適用し，放送型メモリ結合型並列計算機上で実行した結果，プロセッサ台数にほぼ比例した直線的な処理速度の向上が確認された．以上の考察により，動的計画法の高い並列計算性と，示した並列計算アルゴリズムの有効性が確かめられた．

1. 緒　　言

ベルマンにより提案された動的計画法は，原問題を同じ構造をもつ部分問題群に埋め込むことにより，計算機処理に適した再帰的表現を導く．動的計画法は絶対的最適を保証する数少ない数理計画法の一つであり，他の多くの数理計画法と違って，離散変数，制約条件等が概念的には何ら問題を複雑にすることのないすぐれた方法論である[1),2)]．したがって，初期にはORの分野で応用が進められ，後に制御問題，数値計算，パターン認識等々科学技術の諸分野に応用範囲を拡げてきた．

しかしながら，問題の規模が大きくなるに従い著しい計算時間の増大をもたらすので（次元ののろい），進歩の著しい現行の汎用計算機システムの基でも，その有効性を十分に発揮するに至っていないのが現状である．

一方，LSI 技術の著しい発達により，プロセッサの高集積化がその高速化・高機能化を伴い，かつそれが低価格・高信頼度で提供されるようになった．このことにより複数のプロセッサを用いる計算機システムがマイコン革命ともいわれるコンピュータの大衆化路線の延長上で実用化されるようになってきた．さらに多数のプロセッサを有機的に結合する高並列計算処理システムの試行も行われており，大規模行列計算等の高速化が試みられている[3)~9)]．

動的計画法の計算が並列処理可能であれば，それによる処理の高速化が可能となり，動的計画法の潜在的能力を有効に発揮させることができ，動的計画法専用プロセッサの開発も夢ではなくなるであろう．現在まで，定式化することが可能でありながら，その計算時間のゆえに残されていた諸問題を，実際に解くことが可能となる．

以下においては，このような認識の基で，動的計画法の並列処理可能性ならびにその手続きについて述べ，実際に並列計算機上で実行してその有効性を調べる．

2. 直列型動的計画法と表解法

動的計画法が取り扱う最適化過程は，多段決定過程

---

† Parallel Computation Algorithms of Dynamic Programming**** by YASUYUKI SEGUCHI, MASAO TANAKA (Department of Systems Engineering, Faculty of Engineering, Kobe University), TOSHIROH NAKAJIMA (Fundamental Software Division, NEC Corporation), YUKIO KANEDA (Department of Systems Engineering, Faculty of Engineering, Kobe University), MASAKI KOHATA (Okayama University of Science) and SATOSHI NISHINO (Fujisawa Research Institute, IBM Japan Ltd.).

†† 神戸大学工学部システム工学科
††† 日本電気(株)基本ソフトウェア開発本部
†††† 岡山理科大学
††††† 日本アイビーエム(株)藤沢研究所
* 現在　大阪大学基礎工学部機械工学科
** 研究当時　神戸大学工学部システム工学科
*** 研究当時　神戸大学大学院

**** This work was done while Mr. Nakajima and Mr. Nishino were students of Department of Systems Engineering, Kobe University.

であり，状態変換過程を伴う関数再帰方程式で記述される．最も基本的な直列型のものは次のような形式で表される．

$$R_N = r_1 \circ r_2 \circ \cdots \circ r_N \xrightarrow[d_1, d_2, \cdots, d_N]{} \mathrm{opt} \qquad (1)$$

where $r_i = r_i(x_i, d_i)$ $(i=1, \cdots, N)$ (2)

$$x_{i-1} = x_{i-1}(x_i, d_i) \quad (i=1, \cdots, N) \qquad (3)$$

ここで，$\circ$は合成演算子，r_i, x_i は i 段の利得，入力状態である[1]．この決定過程は**図1**に示すような構造をもち，次のような部分問題に書き改められる．

$$f_n(x_n) = \mathop{\mathrm{opt}}_{d_n} Q_n(x_n, d_n) \qquad (4)$$

$$Q_n(x_n, d_n) = \begin{cases} r_n(x_n, d_n) + f_{n-1}(x_{n-1}) & (n \neq 1) \\ r_1(x_1, d_1) \end{cases} \qquad (5)$$

この部分問題群の解 d_n^* は，x_n の関数となっており，原問題への入力状態 x_N を与えると状態変換過程(3)を介して逐次 d_n^* が決定される．

d_n^* を x_n の関数として解析的に求めることは一般に困難であり，微分法等古典的手法が適応できない問題をも取り扱えることも動的計画法の特徴の一つであるから，数値的な解法は動的計画法の重要な部分を占める．代表的な数値解法の一つである表解法(格子法)の求解手順は，**図2**に示すようなもので，前進消去，後退代入の二つの過程からなっている．前者は f_n および d_n^* の作表，後者は表引きに対応している．連続変数に対する格子法の具体的な事項は，ここでの考察の対象とかかわらないので，立ち入らない．

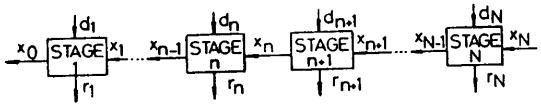


図 1 直列型多段決定過程

Fig. 1 Standard serial dynamic programming problem.

```
for n:=1 to N do begin
 for i:=1 to Kn do begin
  for j:=1 to Lni do begin
   y:=x_{n-1}(x_ni,d_nij);
   Q_n(x_ni,d_nij):=r_n(x_ni,d_nij)∘f_{n-1}(y);
  end j;
  f_n(x_ni):=opt_j Q_n(x_ni,d_nij);
  d_n*(x_ni):=argopt_j Q_n(x_ni,d_nij);
 end i;
end n;
x_N*:=x_N
for n:=N down to 1 do begin
 d_n*:=d_n*(x_n*)
 x_{n-1}*:=x_{n-1}(x_n*,d_n*);
end n;
```

図 2 表解法のアルゴリズム

Fig. 2 Serial processing algorithm for tabular method.

3. 表解法における並列計算性

図2に示した表解法アルゴリズムを吟味することにより，動的計画法の並列計算可能性について検討する．

f_n および d_n^* の作表過程は3重の繰り返し，段，入力状態，決定についての繰り返し処理から構成されている．最も内側の決定についての繰り返しに注目する．ある状態 x_n について可能な決定 $d_n \in \boldsymbol{D}_n(x_n)$ について，Q_n を求めるに必要な情報は，$x_{n-1} = x_{n-1}(x_n, d_n)$, $f_{n-1}(x_{n-1})$, $r_n(x_n, d_n)$ であり，いま求めている Q_n やそれから得られる f_n は必要でない．したがってすべての可能な d_n について並列的に Q_n を求めることができる．x_n についての Q_n が得られた後の $f_n(x_n)$, $d_n^*(x_n)$ を求める処理は，2進木を構成して探索を行えば並列的に実行可能であり，これはすべての d_n について $Q_n(x_n, d_n)$ を並列的に求める効果を十分有効に利用している．

入力状態についての繰返しは，上で検討した $f_n(x_n)$, $d_n^*(x_n)$ を求める計算を，すべての $x_n \in \boldsymbol{X}_n$ について行うためのもので，各 x_n についての情報は先とまったく同じであり，得ようとしている f_n および d_n^* を必要としない．したがって，すべての x_n について並列処理可能である．これら二つの繰返しは，本質的に逐次的なものではなく，計算機の逐次処理に由来したものであり，これらの処理は高い並列性を有していることがわかる．この並列性を考慮すれば，**図3**に示すようなアルゴリズムが得られる．

次に，外側の段についての繰返しについて考える．直列型多段決定過程における情報の流れは**図4**に示さ

```
for n:=1 to N do begin
 forall x_n∈ x_n do in parallel begin
  forall d_n∈ d_n(x_n) do in parallel begin
   y:=x_{n-1}(x_n,d_n);
   Q_n(x_n,d_n):=r_n(x_n,d_n)∘f_{n-1}(y);
  end d_n;
 f_n(x_n):=opt_{d_n(x_n)} Q_n(x_n,d_n);
 d_n*(x_n):=argopt_{d_n(x_n)} Q_n(x_n,d_n);
 end x_n;
end n;
```

図 3 前進（作表）過程の並列計算アルゴリズム（状態と決定に注目した場合）

Fig. 3 Parallel processing algorithm for forward tabular process in terms of "state" and "decision".

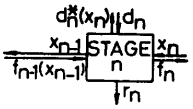


図 4 n段における情報の流れ
Fig. 4 Data flow at the stage n.

```
forall n∈[1,...,N] do in parallel begin
 forall x_n∈X_n do in parallel begin
  forall d_n∈D_n(x_n) do in parallel begin
   x_{n-1}:=x_{n-1}(x_n,d_n);
   receive(f_{n-1}(x_{n-1}));
   Q_n(x_n,d_n):=r_n(x_n,d_n)∘f_{n-1}(x_{n-1});
  end d_n;
  f_n(x_n):=opt_{d_n(x_n)} Q_n(x_n,d_n);
  d_n*(x_n):=argopt_{d_n(x_n)} Q_n(x_n,d_n);
  send(f_n(x_n));
 end x_n;
end n;
```

図 5 前進(作表)過程の並列計算アルゴリズム
(部分利得関数のタイミングをとる場合)
Fig. 5 Parallel processing algorithm for forward tabular process with emphasis on the calculation of subsystem return.

れる．すなわち，1から $n-1$ 段の影響が部分利得 $f_{n-1}(x_{n-1})$ に集約された後に n 段での $f_n(x_n)$ を求めるのが直列型の直列性である．しかしながらもう少し詳細に検討すると，これは f_n をひとまとめにした直列性を意味していないことがわかる．

ある x_n と d_n の組について Q_n を求めるに必要な f_{n-1} は，状態変換された $x_{n-1}=x_{n-1}(x_n,d_n)$ に関する $f_{n-1}(x_{n-1})$ のみである．すなわち，ある x_n について，すべての $x_{n-1}\in\{x_{n-1}(x_n,d_n)\mid d_n\in D_n(x_n)\}$ の f_{n-1} が得られていれば，$f_n(x_n), d_n^*(x_n)$ は計算可能である．この情報の連結が処理できれば，段についての繰返しも完全な直列型である必要はない．この考え方に基づくアルゴリズムとしては，図5に示すようなものが考えられる．ここで，receive 文は引数に値が与えられるまで待機することを，send 文は引数の値を receive 文に告知することを意味する．

以上概観したように，DP の表解法は繰返し処理で表現されるのが常であるが，これは逐次処理型計算機によるもので，その大部分は並列的に処理しうることがわかった．

4. 部分問題への分解過程と並列計算性

前章では，通常の表解法アルゴリズムを機械的に検討し，その並列計算性を示したが，ここでは部分問題への分解過程に注目しながら，DP の並列計算性について検討する．

原問題(1)の2段決定過程への分解を考える．1から n 段と $n+1$ から N 段をそれぞれⅠ，Ⅱで示し，利得 R_N をⅠ，Ⅱに対応して分けると，次式のように書ける．

$$R_N=R_{\mathrm{I}}\circ R_{\mathrm{II}} \tag{6}$$

$$R_{\mathrm{I}}=r_1\circ r_2\circ\cdots\circ r_n,\quad R_{\mathrm{II}}=r_{n+1}\circ r_{n+2}\circ\cdots\circ r_N \tag{7}$$

状態変換(3)を適当に用いれば，2段決定過程

$$\left.\begin{aligned}&\mathrm{opt}_{d_{\mathrm{I}},d_{\mathrm{II}}}R_N=\mathrm{opt}_{d_{\mathrm{II}}}\{R_{\mathrm{II}}(x_{\mathrm{II}},d_{\mathrm{II}})\circ\hat{R}_{\mathrm{I}}(x_{\mathrm{I}})\}\\&\hat{R}_{\mathrm{I}}(x_{\mathrm{I}})=\mathrm{opt}_{d_{\mathrm{I}}}R_{\mathrm{I}}(x_{\mathrm{I}},d_{\mathrm{I}})\end{aligned}\right\} \tag{8}$$

$$\left.\begin{aligned}&d_{\mathrm{I}}=(d_1,d_2,\cdots,d_n),\quad d_{\mathrm{II}}=(d_{n+1},\cdots,d_N)\\&x_{\mathrm{I}}=x_{\mathrm{I}}(x_{\mathrm{II}},d_{\mathrm{II}})=x_n,\quad x_{\mathrm{II}}=x_N\end{aligned}\right\} \tag{9}$$

が得られる．(8)を x_{I} に注目して書き改めると，

$$\mathrm{opt}_{d_{\mathrm{I}},d_{\mathrm{II}}}R_N=\mathrm{opt}_{x_{\mathrm{I}}}\{\hat{R}_{\mathrm{II}}(x_{\mathrm{I}})\circ\hat{R}_{\mathrm{I}}(x_{\mathrm{I}})\} \tag{10a}$$

$$\hat{R}_{\mathrm{II}}(x_{\mathrm{I}})=\mathrm{opt}_{d_{\mathrm{II}}\in\{d_{\mathrm{II}}\mid x_{\mathrm{I}}=x_{\mathrm{I}}(x_{\mathrm{II}},d_{\mathrm{II}})\}}R_{\mathrm{II}}(x_{\mathrm{II}},d_{\mathrm{II}}) \tag{10b}$$

$$\hat{R}_{\mathrm{I}}(x_{\mathrm{I}})=\mathrm{opt}_{d_{\mathrm{I}}}R_{\mathrm{I}}(x_{\mathrm{I}},d_{\mathrm{I}}) \tag{10c}$$

となり，式(10b)，(10c)は，$N-n$ 段および n 段決定過程であり，これらを統合して原問題の解を求める過程が(10a)である．式(10b)，(10c)はすべての中間状態 x_{I} について解かれねばならないが，異なる x_{I} についてのそれらは独立しており，並列処理が可能である．また，ある x_{I} に注目した場合，$N-n$ 段および n 段決定は独立したものである．したがって，式(10b)，(10c)を並列にかつ，すべての x_{I} についてもそれらを並列に処理することが可能である．これらを，x_{I} をキーとして統合する(10a)は2進木を構成して行えば，やはり並列的に処理できる．この様子を，図6に示す．それぞれの部分多段決定過程をどのようにして解くかは任意である．

5. 反復的解法と並列計算性

動的計画法が最も有効に働くのは直列型多段決定問

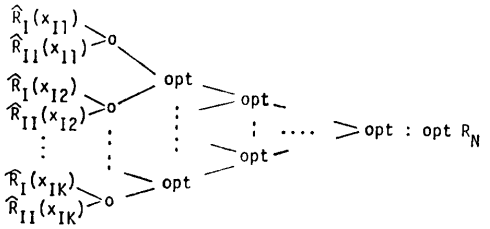


図 6 部分問題への分解に注目した並列計算過程
Fig. 6 Parallel processing scheme based on subproblem decomposition.

題であるが，非直列型問題に関しての解法についての考察も進められ，その守備範囲は著しく拡大している．多くの問題は多段決定過程として定式化可能であるが，規模や非直列性のゆえに実際に解くのが困難である場合が少なくない．このような問題の解を求めるために，発見的解法の研究が進められている[10]．ここでは，その一つである反復的解法をとりあげ，並列計算性について考察する．

5.1 反復的解法

非常に簡単ではあるが，(1)とはやや，異なる次のような多段決定過程を例にとりあげる．

$$R_N = r_1 \circ r_2 \circ \cdots \circ r_N \xrightarrow[d_1, \cdots, d_{N+1}]{} \mathrm{opt} \tag{11}$$

$$r_n = r_n(x_n, d_n, d_{n+1}) \tag{12}$$

$$x_{n-1} = x_{n-1}(x_n, d_n) \tag{13}$$

問題(1)と異なるのは，利得 r_n に隣接段での決定 d_{n+1} が影響していることである．問題(11)の多段決定表現は，

$$f_{N+1}(x_N) = \mathrm{opt}_{d_{N+1}} f_N(x_N, d_{N+1}) \tag{14a}$$

$$f_n(x_n, d_{n+1}) = \mathrm{opt}_{d_n}[r_n(x_n, d_n, d_{n+1}) \circ f_{n-1}(x_{n-1}, d_n)] \tag{14b}$$

$$f_1(x_1, d_2) = \mathrm{opt}_{d_1} r_1(x_1, d_1, d_2) \tag{14c}$$

であり，その構造は**図7**に示すようなものである．これは x_n と d_{n+1} の対を新たに状態と定めれば，(1)と同じ取扱いが可能であるが，ここでは，x_n を状態，d_{n+1} を非直列効果として取り扱う．

非直列効果を推定値で置換えることで切断し，各段相互の関連を直列型のそれのみにする反復解法は，

$$\left.\begin{aligned} f_{N+1}(x_N) &= \mathrm{opt}_{d_{N+1}}[r_N(x_N, d_N^{(l)}, d_{N+1}) \circ f_{N-1}(x_{N-1}, d_N^{(l)})] \\ f_n(x_n, d_{n+1}^{(l)}) &= \mathrm{opt}_{d_n}[r_n(x_n, d_n, d_{n+1}^{(l)}) \circ f_{n-1}(x_{n-1}, d_n)] \\ f_1(x_1, d_2^{(l)}) &= \mathrm{opt}_{d_1} r_1(x_1, d_1, d_2^{(l)}) \end{aligned}\right\} \tag{15}$$

であり，$d_i^{(l)}$ は決定の推定値である．この様子を**図8**に示す．用いた推定値と得られたそれとが一致すれば，切断された非直列効果が回復されたことになる．

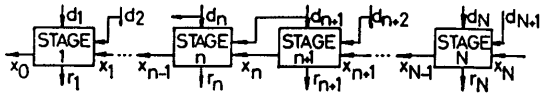


図 7 やや複雑な多段決定過程
Fig. 7 A sophistication of serial dynamic programming problem.

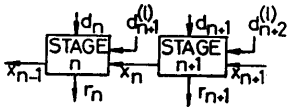


図 8 直列型への簡略化
Fig. 8 Reduction into standard serial problem.

5.2 反復的解法の並列計算性

問題(14)の各反復は直列型多段決定過程であり，その前進過程は2.2節の考察より並列計算可能であるが，後退過程は直列処理されねばならないので，このままでは，2.2節以上の並列処理は得られない．そこで，別の発見的手法である状態空間を制限する反復解法との組合せについて考える．

(15)において状態 x_n を現在の推定値 $d_i^{(l)}{}_{i=1,\cdots,N+1}$ により，式(13)で変換された $x_n^{(l)}$ に制限すれば，前進過程だけで新たな推定 $d_n^{(l+1)}$ が定まり，後退過程がなくなる．

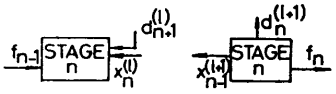
$$d_n^{(l+1)} = \mathrm{argopt}_{d_n}[r_n(x_n^{(l)}, d_n, d_{n+1}^{(l)}) + f_{n-1}(x_{n-1}(x_n^{(l)}, d_n), d_n^{(l)})] \tag{16}$$

ここで，x_{n-1} および f_n は次のように定める．

$$x_{n-1}^{(l+1)} = x_{n-1}(x_n^{(l)}, d_n^{(l+1)}) \tag{17}$$

$$f_n(x_n, d_{n+1}^{(l)}) = r_n(x_n, d_n^{(l+1)}, d_{n+1}^{(l)}) + f_{n-1}(x_{n-1}(x_n, d_n^{(l+1)}), d_n^{(l+1)}) \tag{18}$$

これは並列計算の面からは，非常に有利に働く．$d_n^{(l)}$ を求めるには f_{n-1} ならびに $d_{n+1}^{(l-1)}$ が必要であ



(a) $(l+1)$ 推定時 (b) $(l+1)$ 推定後
(a) Before $(l+1)$th estimation. (b) After $(l+1)$th estimation.

図 9 反復解法の情報の流れ
Fig. 9 Data flow for iterative method.

```
repeat in parallel
 forall n ∈ [1,...,N] do in parallel begin
  receive (x_n, d_n+1);
  forall d_n ∈ d_n(x_n) do in parallel begin
   y:=x_n-1(x_n, d_n);
   receive (f_n-1(y));
   Q_n(d_n):=r_n(x_n, d_n, d_n+1) ∘ f_n-1(y);
  end d_n;
  d_n:=argopt_{d_n(x_n)} Q_n(d_n);
  x_n-1:=x_n-1(x_n, d_n);
  send (x_n-1, d_n);
  forall y ∈ x_n do in parallel begin
   z:=x_n-1(y, d_n);
   f_n(y):=r_n(y, d_n, d_n+1) ∘ f_n-1(z);
   send (f_n(y));
  end y;
 end n;
until converged;
```

図 10 反復解法の並列計算アルゴリズム
Fig. 10 Parallel processing algorithm for iterative method.

り，$d_n^{(i)}$ が得られた後，$n-1$ 段に $d_n^{(i)}$ を，$n+1$ 段には f_n を供給する．これを**図9**に示す．n 段で $d_n^{(i)}$ が求まったとき，$n-1$段で $d_{n+1}^{(i-1)}$ を，$n+1$ 段で $d_{n+1}^{(i)}$ を求めるに必要な情報が整う．このことは，段についての直列的繰り返しと，反復解法による繰り返しが結合できることを示している．この点に注目すれば，**図10**に示すような並列型のアルゴリズムが考えられ，これを情報の流れで示したものが**図11**である．これより，図9に示した情報の入出力と，段および反復の繰り返しとが調和していることがわかる．

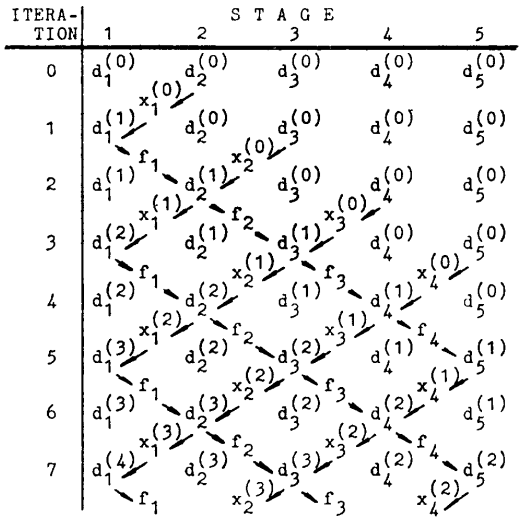
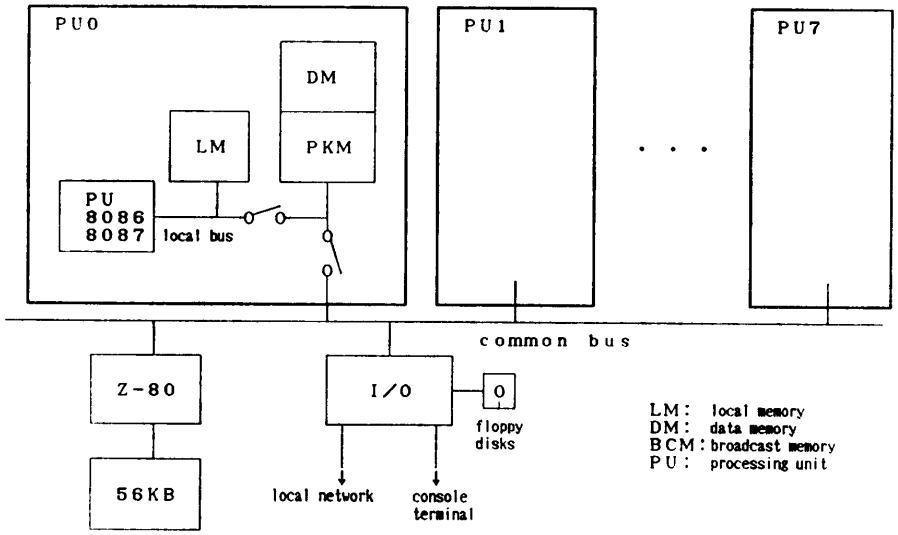


図 11 反復解法の並列計算時の情報の流れ
Fig. 11 Data flow diagram of parallel iterative algorithm.

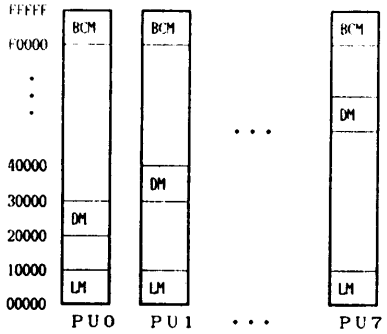
6. BCM 結合型並列計算機での実行

この章では，提案した動的計画法の並列計算アルゴリズムのいくつかを，著者らの一部が開発したブロードキャストメモリ (BCM) 結合型並列計算機[11]上で実行した結果を示し，その効果について述べる．

用いる計算機システムのハードウェア構成，メモリ



(a) ハードウェア構成
(a) Hardware system.



(b) メモリ構成
(b) Memory system.

図 12 ブロードキャストメモリ結合型並列計算機システム
Fig. 12 Broadcast memory connected parallel processor.

構成を図12に示す．プロセッサはインテル社の16ビットプロセッサ8086に数値演算プロセッサ8087を付加したマルチプロセッサ8ユニットからなっている．メモリシステムは，各プロセッサユニットに固有のローカルメモリ，他のユニットからもバスを通してアクセスされるデータメモリ，ならびに書込み時に全ユニットの同アドレスに同じデータが自動的に放送転送されるブロードキャストメモリからなっている．ソフトウェアシステムとしては，OSにCPM/86を用いており，並列計算プログラムは，Cを拡張した並列プログラミング言語Q[12]を用いて記述される．使用システムの詳細は，文献11),12)に述べられているので省略する．

まずはじめに，動的計画法の代表的問題である配分問題[13]を例にとり，表解法の並列計算アルゴリズムを適用する．

$$\sum_{i=1}^{N} r_i \xrightarrow[d_1,\cdots,d_N]{} \min$$

where $r_i = r_i(d_i) = a_i d_i^2, \quad a_i > 0$

$$x_{i-1} = x_i - d_i, \qquad d_1 + \cdots + d_N = X_N$$

用いるアルゴリズムは図3に示したものである．図13は，段数 $N=10$，入力状態 $X_N=100$ としたときの処理速度の向上を示したものである．プロセッサ1台による直列処理速度に比べ，プロセッサ4台で3.92倍，8台で7.6倍の速度向上を示しており，プロセッサ台数 m に対して $0.95m$ 以上のスピードアップが達成されている．台数増加につれて若干の低下はみられるが，ほぼ線形の速度向上となっている．

次に，三重ブロック対角行列を係数とする一次方程式の動的計画法による解法[13),14)]を例にとり，反復解法の並列計算を行う．この問題は，二次形式の和の最小化問題として記述される．

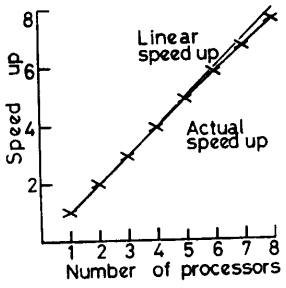


図13 表解法の並列計算による速度向上（配分問題）
Fig. 13 Speed-up by parallel tabular algorithm in terms of state and decision—in case of allocation problem.

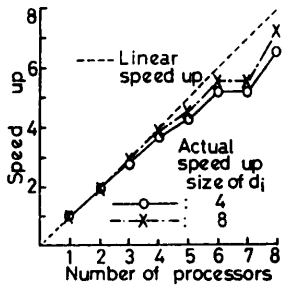


図14 反復的解法の並列計算による速度向上（三重ブロック対角係数行列一次方程式の動的計画法による解法）
Fig. 14 Speed-up by parallel iterative algorithm —DP formulation based solution of block-tridiagonal linear system.

$$\sum_{i=1}^{N} r_i \xrightarrow[d_1,\cdots,d_{N+1}]{} \min$$

where $r_i(d_i, d_{i+1})$

$$= (d_i^T, d_{i+1}^T)\left\{\begin{pmatrix} A_i^{11} & A_i^{12} \\ A_i^{21} & A_i^{22} \end{pmatrix}\begin{pmatrix} d_i \\ d_{i+1} \end{pmatrix} - 2\begin{pmatrix} b_i^1 \\ b_i^2 \end{pmatrix}\right\}$$

未知の決定変数 d_i はベクトル，既知の係数 A_i, b_i はマトリクス，ベクトルである．各部分問題(16)の解法には共役勾配法を用いる．図14は段数 $N=24$，決定ベクトル d_i の次元が4および8のときの処理速度を示したものである．プロセッサが5,7台の場合を除いてほぼ直線的速度向上となっている．これは，この問題では各部分問題の規模が等しく，反復ごとに同期をとったため，段についての24の並列処理可能な負荷を，5, 7台のプロセッサに等しく配分できなかったためである．また，d_i の次元が大きいと，部分問題の規模が大きくなり，オーバヘッドが相対的に減少し，処理速度の向上が顕著である．

7. 結　　言

動的計画法により明確な定式化が可能でありながら，実際に解きうる問題が計算機の能力のゆえに大きく制約されていることは，その利用者にとって重大な問題である．本研究においては，並列計算方式を採用した動的計画法計算の高速化のための基礎として，動的計画法の並列計算性について検討し，いくつかの並列計算アルゴリズムを示した．

表解法における繰り返し計算のうち，決定変数ならびに状態変数についてのそれは，並列処理が可能であり，段についてのそれは，直列型ではあるが，部分利得関数の計算とのタイミングを計れば，必ずしも完全な直列型ではないことが明らかとなった．

動的計画問題として定式化を行う際の部分問題群への分解過程に注目し，小規模な動的計画問題を並列して解いた後，それらを関連付ける中間状態をキーとして原問題の解を得ることにより，表解法とは異なる視点から，動的計画法計算の並列処理性を示した．

さらに，直列型多段決定過程よりも複雑な構造をとる多段決定過程を，直列型問題を反復して解くことで発見的に解を得る反復的解法について検討を加えた．その結果，状態変換過程を制限する方法と組み合わせることにより，直列型問題の段についての繰り返しと，解法自身の反復繰り返しとを結合することが可能となり，並列計算に適した構造をもつことが明らかとなった．

これらの考察は，動的計画法の取り扱う直列型多段決定過程の直列型という用語とはうらはらに，高い並列計算可能性を有していることを示している．また，提案したアルゴリズムのいくつかを，簡単な例題についてBCM結合型並列計算機上で実行した結果，直列計算に比べてプロセッサ台数にほぼ比例する処理速度の向上が確かめられ，提案したアルゴリズムの有効性を示すことができた．

ここで示した並列計算性を，実際的な問題の並列計算を実行する上において，どのように選択，組み合わせるか，またそれらのより詳細な評価は今後の課題となろう．

参考文献

1) 鍋島：動的計画法，森北出版，東京 (1968).
2) Nemhauser, G. L.: *Introduction to Dynamic Programming*, John Wiley & Sons, New York (1966).
3) 金田：並列処理システムによる連立一次方程式と楕円形偏微分方程式の数値計算法，情報処理，Vol. 16, No. 2, pp. 122-129 (1975).
4) 金田：並列処理システムによる線形計画計算と実対称行列の三重対角化計算，情報処理，Vol. 19, No. 1, pp. 39-45 (1978).
5) 金田：環状結合型超多重プロセッサシステムによる大次元連立一次方程式の並列計算，情報処理学会論文誌，Vol. 21, No. 5, pp. 402-406 (1980).
6) 金田，小畑，前川：マトリックスブロードキャストメモリ結合形並列計算機による n 元連立一次方程式の $O(n)$ 時間計算，情報処理学会論文誌，Vol. 23, No. 5, pp. 570-575 (1982).
7) 小山，牧野，三木，石動，飯野，井関：パラレル・プロセッサ・アレイを用いた連続系シミュレータのアーキテクチャ，シミュレーション，Vol. 1, No. 1, pp. 42-49 (1981).
8) 土肥，小山：有限要素法の並列処理手法，電子通信学会論文誌，Vol. J 65-D, No. 4, pp. 464-470 (1982).
9) 金田，小畑，前川：BC プロセッサアレイと高並列マトリクス計算，情報処理学会論文誌，Vol. 24, No. 2, pp. 175-180 (1983).
10) Norman, J. M.: *Heuristic Procedure in Dynamic Programming*, Manchester Univ. Press, Manchester (1972).
11) 小畑，金田，前川：ブロードキャストメモリ結合型マルチマイクロプロセッサシステムの試作，情報処理学会論文誌，Vol. 24, No. 5, pp. 351-356 (1983).
12) 小畑，金田，田中，前川：並列プログラミング言語の設計と実現，情報処理学会第45回計算機アーキテクチャ研究会資料，53, pp. 43-48 (1984. 5).
13) 杉山：動的計画法，日科技連，東京 (1976).
14) Distéfano, N. and Samartin, A.: A Dynamic Programming Approach to the Formulation and Solution of Finite Element Equations, *Comput. Method Appl. Mech. Eng.*, Vol. 5, No. 1, pp. 37-52 (1975).

(昭和59年9月28日受付)
(昭和60年2月21日採録)