ソフトウェア基礎論　23－6
（1987. 12. 10）

プログラム群を並行処理するシステムの一評価方式

PERFORMANCE EVALUATION OF SYSTEMS FOR PROGRAMS WITH VARIATION IN CONCURRENCY

J．チャニントーン †　渡辺　尚 ††　中西　暉 †　手塚　慶一 †
J.CHANINTORN †　T.WATANABE ††　H.NAKANISHI †　Y.TEZUKA †

† 大阪大学工学部通信工学科
† Faculty of Engineering, Osaka University

†† 徳島大学工学部情報工学科
†† Faculty of Engineering, Tokushima University

あらまし　並行処理システムの性能評価のために、従来並列度が固定されたプログラム群を対象にした解析が発表されている。本研究では、異なる並列度を持ちしかも並列度が時間と共に変化するより一般的なプログラム群を対象にするため、分解近似手法に平均並列度を導入した近似手法を提案する。本手法は計算機時間の短縮・必要メモリ量の軽減という利点を持つ。いくつかのモデルについて本手法で得られた結果と大型計算機上で実行したシミュレーションの結果とを比較し、その有効性を検討する。

Abstract　Decomposition approximation, based upon the mean value analysis algorithm, for programs with fixed concurrency has been recently proposed. Using queueing network models, this paper presents an approximation method to model the computer systems for programs with variation in concurrency. The allowance of variation in concurrency for each program provides available network conditions for an improvement of network flexibility. We also show that our method is faster than the one proposed previously from the viewpoint of memory space requirements. Using examples, the accuracy of our method is found by checking results against those of the more exact simulations.

1．まえがき

近年、複数のプログラムを並行処理するシステムが盛んに研究されている。これらのシステムの効率改善効果、ボトルネック等の検討には、計算機を用いたシミュレーションと並んで、待ち行列網理論に基づいた近似解析が行われている[1]~[10]。

特に［5］では、平均値解析を利用した分解近似と呼ばれる解析手法を提案し、この解析精度の高さを実証している。［5］では、すべてのプログラムが持つ並列度は同一であり、また時間に対して固定されているものだけを対象にしている。ＴＳＳのように、１つのシステムを多人数で用いている場合には、平均的に見て並列度を固定しても誤差はさほど問題にならない。しかし、今日のようにワークステーションを少人数で共用するなど計算機が準パーソナル化している状況では、それぞれのプログラムの特性を解析に反映させる必要がある。

以上のような観点から、本研究では各個人からジョブとして投入されるプログラムの並列度が各プログラム毎に異なったり、時間と共に変化するようなプログラム群を並行処理するシステムの性能評価を目的とする。

プログラム群は、以下の３つのタイプを考える。

（１）　プログラム毎に並列性が異なるプログラム群（ＤＣＰ）
（２　時間と共に並列性が変化するプログラム群（ＴＣＰ）
（３）　（１）（２）両者の性質を持ったプログラム群（ＶＣＰ）

ＤＣＰに対する解析は、分解近似によっても可能でありまた文献［3］でも試みられているが、プログラム数の増加とともに急速に状態数が多くなり、実質的には困難である。また、ＴＣＰ、ＶＣＰに対しては従来の手法では取り扱えない。

本研究では、まずタスク間の同期を取る間隔をサイクルと考えサイクル毎の並列度を要素とする並列度ベクトルを提案する。次に、並列度ベクトルをもとにした平均並列度を用いて上の３種のプログラム群を並行処理するシステムの性能評価法を考察する。

2．解析モデル

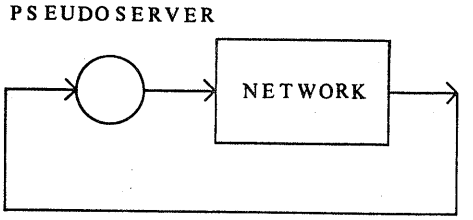

図1．解析モデル

ここで、解析モデルを以上のように仮定する。プログラム群は完全に独立なNコのプログラムによって構成されており、それぞれのプログラムを一次タスクと呼ぶ。本研究で考える並行処理システムは図1．に示すような複数のサーバ（CPUやI／O DEVICEなど）からなるネットワークと疑似サーバを含んでいる。一次タスクはまず、サービス時間なしの疑似サーバに入り、いくつかの二次タスクに分割される。同一の一次タスクから生まれた二次タスクをシブリングと呼ぶ。分割された二次タスクはネットワークによって実行される。二次タスク間にはネットワーク内での待ち行列の影響を除いてデータ依存等の関係はないものとし、二次タスクが訪れるサーバはマルコフ連鎖を成すと仮定する。並行処理システムにおいて二次タスクの実行が終了した場合、疑似サーバのバッファに戻される。この時シブリングの実行が終っていない場合、このバッファで待ち合わせが行われる。全部のシブリングの実行が終ると一次タスクに戻り、再び疑似サーバに入る。本システムは以上の過程が繰り返される。尚、一次タスクが疑似サーバに入ってから再び疑似サーバに戻るまでをサイクルと呼び、1サイクルに費やした時間をサイクルタイムと定義する。また、一次タスク（識別子j）が、疑似サーバにおいてXjコの二次タスクに分割できる時、この二次タスクは並列度Xjを持つという。一般に、Xjはサイクル毎に変化する。 そこで、サイクルごとの並列度をベクトル $C_j = \{ X_{j1}, X_{j2}, X_{j3}, \ldots \}$ と表し、これを並列度ベクトルと呼ぶ。

3．ソフトウェアモデル

3．1 DCP (Different Concurrent Programs)

並行処理されるプログラム群について考えるとプログラム毎に並列度が固定されているとは限られない。そこでまず、プログラム毎に並列度ベクトルが異なる場合を考える。ただし、並列度はサイクルによって変化しない。例としてDCPを実行している状態を図2．に示す。

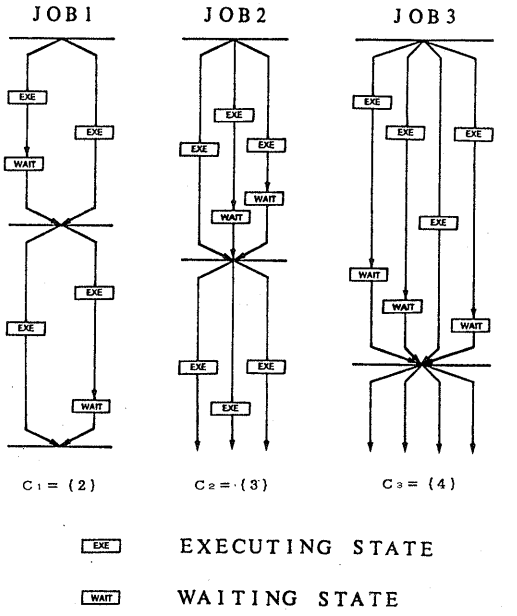

図2．DCPを実行している状態

3．2 TCP (Time-dependent Concurrent programs)

TCPを実行している状態を図3．に示す。TCPはサイクル毎に並列度が変化するプログラム群である。ただし、 プログラム毎に並列度ベクトルは同一である。TCPを処理している状態はシミュレーション等でよく見られるような1つのプログラムをコピーし、それぞれに対して異なるデータファイルを与えて実行している場合に相当する。

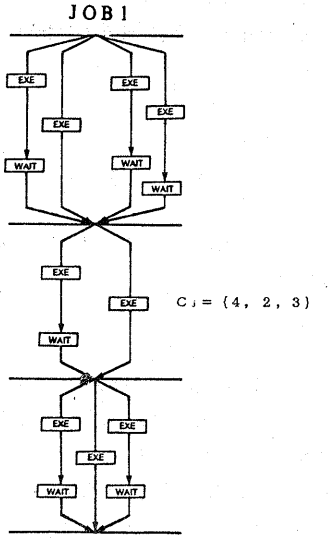

図3．ＴＣＰを実行している状態

3．3 ＶＣＰ (Variable Concurrent Programs)

ＶＣＰはＤＣＰとＴＣＰを混合したものであり、最も一般的な形である。すなわち、並列度ベクトルはサイクルとプログラム毎に変化する。ＶＣＰを実行している状態を図4．に示す。

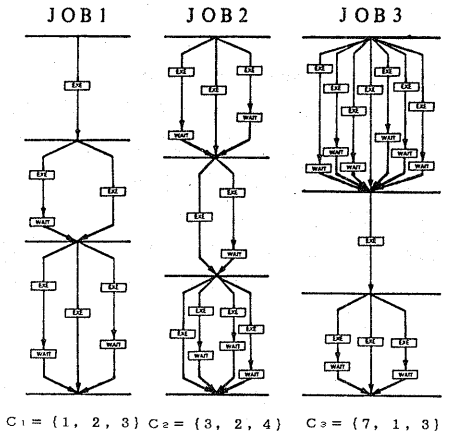

図4．ＶＣＰを実行している状態

4．提案方式

4．1 近似手法の概要

本研究で提案する近似手法ではまずプログラム群の持つ平均並列度$\bar{C}$を計算する。そして$\bar{C}$をはさむ整数$\bar{C}_1$、$\bar{C}_2$（$=\bar{C}_1+1$）に対して分解近似によって特性値（スループット、待ち行列長、利用率）を求める。次に$\bar{C}$の$\bar{C}_1$からの偏差を基にした比例配分によって$\bar{C}$に対する特性値を求める。

4．2 平均並列度の計算

以下のように記号を定義する。

Ｎ ： 一次タスクの数、

ｊ ： 一次タスクの識別子（ｊ＝1,2,...,N)、

Ｃｊ： 一次タスク ｊ の並列度ベクトル、

Ｍｊ： Ｃｊの要素の個数、

Ｘｊｉ： 一次タスク ｊの ｉ 番目のサイクルにおける並列度、

Ｔｊｉ： 一次タスク ｊ のサイクル ｉ で費やした時間、

$\bar{C}$ ： 二次タスクの平均並列度

とする。$\bar{C}$はＤＣＰ、ＴＣＰ、ＶＣＰに対してそれぞれ次式により計算できる。

4．2．1 ＤＣＰの場合

$$C_1 = \{X_{11}\}$$
$$C_2 = \{X_{21}\}$$
$$C_3 = \{X_{31}\}$$
$$\cdot$$
$$\cdot$$
$$\cdot$$
$$C_j = \{X_{j1}\}$$
$$j = 1, 2, 3, \ldots, N$$

$$\bar{C} \equiv \left(\sum_{j=1}^{N} X_{j1}\right) \Big/ \left(\sum_{j=1}^{N} M_j\right)$$

$$\equiv (1/N) \sum_{j=1}^{N} X_{j1} \qquad (1)$$

4．2．2 TCPの場合

$$C_1=C_2=C_3=\cdots=C_j$$
$$=\{X_{11}, X_{12}, X_{13}, \ldots, X_{1M_1}\}$$

$$\bar{C}\equiv \left(\sum_{i=1}^{M_1} T_{1i}X_{1i}\right) / \left(\sum_{i=1}^{M_1} T_{1i}\right) \qquad (2)$$

ここで、一次タスク1のサイクル i でのサイクルタイムを T_{1i}とする。全てのiに対してT_{1i}＝定数の場合、式（2）は次式に簡単化される。

$$\bar{C}\equiv (1/M_1)\sum_{i=1}^{M_1} X_{1i} \qquad (2')$$

4．2．3 VCPの場合

$$C_1=\{X_{11}, X_{12}, X_{13}, \ldots, X_{1i}\}$$
$$C_2=\{X_{21}, X_{22}, X_{23}, \ldots, X_{2i}\}$$
$$C_3=\{X_{31}, X_{32}, X_{33}, \ldots, X_{3i}\}$$
$$\vdots \qquad\qquad \vdots$$
$$C_j=\{X_{j1}, X_{j2}, X_{j3}, \ldots, X_{ji}\}$$
$$i=1, 2, 3, \ldots, Mj$$
$$j=1, 2, 3, \ldots, N$$

$$\bar{C}\equiv \left(\sum_{j=1}^{N}\sum_{i=1}^{Mj} T_{ji}X_{ji}\right) / \left(\sum_{j=1}^{N}\sum_{i=1}^{Mj} T_{ji}\right) \qquad (3)$$

全てのiに対して T_{1i}＝定数の場合、式（3）は次式のように表せる。

$$\bar{C}\equiv \left(\sum_{j=1}^{N}\sum_{i=1}^{Mj} X_{ji}\right) / \left(\sum_{j=1}^{N} Mj\right) \qquad (3')$$

4．3 $\bar{C}$の分解

一般に非整数の平均並列度$\bar{C}$を整数部Ｉと小数部Ｆに分ける。

$$\bar{C}=I+F=(1-F)\,I+F\,(I+1)$$
$$=(1-F)\,\bar{C}_1+F\,\bar{C}_2 \qquad (4)$$

そして並列度が$\bar{C}_1$と$\bar{C}_2$のそれぞれの場合に対して以下の分解近似を行う。

4．3．1 分解近似

ここでは分解近似[5]の概略を述べる。並列度がＫである二次タスクを処理している場合、まず、 時刻ｔにおいて実行されているタイプ ｍ（0≦m≦K：整数）の二次タスクの数を$\vec{a}_m(t)$とすると二次タスクの実行状態を表すベクトル $\vec{a}$は$[a_0(t), a_1(t), a_2(t), \ldots, a_K(t)]$で表される。

状態 $\vec{a}$の遷移速度が指数分布的に変化すると仮定すると全ての$\vec{a}$に対して状態遷移速度行列Ｑ（$\vec{a}$）を計算することができる。Ｑ（$\vec{a}$）が解けると全ての$\vec{a}$に対して状態$\vec{a}$の確率行列 ｐ（$\vec{a}$）は以下の平衡方程式[12]により計算できる。

$$p(\vec{a})\,Q(\vec{a})=0 \qquad (5)$$
$$\sum_{\vec{a}} p(\vec{a})=1 \qquad (6)$$

さらに、

ρｎ（$\vec{a}$）：状態$\vec{a}$のサーバｎの利用率、

Ｌｎ（$\vec{a}$）：状態$\vec{a}$のサーバｎの平均待ち行列長、

λ_0（$\vec{a}$）：状態$\vec{a}$のサーバ０での一次タスクのスループットとすれば、これらを平均値解析[11]により、全ての$\vec{a}$に対して計算すれば、並列度Ｋの時のネットワーク全体のサーバｎの利用率 ρｎ（Ｋ）、 待ち行列長Ｌｎ（Ｋ）、サーバ０でのスループットλ_0（Ｋ）は次式によって得られる。

$$\rho n(K)=\sum_{\vec{a}} p(\vec{a})\,\rho n(\vec{a}) \qquad (7)$$

$$Ln(K)=\sum_{\vec{a}} p(\vec{a})\,Ln(\vec{a}) \qquad (8)$$

$$\lambda_0(K)=\sum_{\vec{a}} p(\vec{a})\,\lambda_0(\vec{a}) \qquad (9)$$

4．3．2 $\bar{C}$の比例配分

4．3．1においてK＝$\bar{C}_1$と$\bar{C}_2$の場合について計算した特性値から、 $\bar{C}$ に対する特性値 $\rho n(\bar{C})$、$Ln(\bar{C})$、$\lambda_0(\bar{C})$を

$$\rho n(\bar{C}) = (1-F)\,\rho n(\bar{C}_1) + (F)\,\rho n(\bar{C}_2) \quad (10)$$

$$Ln(\bar{C}) = (1-F)\,Ln(\bar{C}_1) + (F)\,Ln(\bar{C}_2) \quad (11)$$

$$\lambda_0(\bar{C}) = (1-F)\,\lambda_0(\bar{C}_1) + (F)\,\lambda_0(\bar{C}_2) \quad (12)$$

として計算する。

以上のアルゴリズムをプログラミング言語によって表現したものが図5．である。

```
/******ALGORITHM******/

BEGIN  /**Computation for C̄**/

INPUT  parameters
: Cj(concurrency vector of j-th primary task)
: N (number of all primary tasks)
: Mj(number of total elements of Cj)

IF  Cj  is represented by the function of j
    (DCP) only.
  THEN

                  N
    C̄ ≡ (1/N) Σ  Xj1                    (1)
                 j=1

  ELSE  IF  Cj  is represented by the
        function of Tij  (TCP) only.

                  M1
    C̄ ≡ (1/M1) Σ  X1i                   (2')
                 i=1

        ELSE  it is VCP.

                  N  Mj          N
    C̄ ≡ (Σ Σ Xji) / (Σ Mj)     (3')
               j=1 i=1        j=1

        END  IF
END  IF
END  /*****C̄ is solved*****/


BEGIN  /**Initialize C̄1 and C̄2**
   ** Let I and F be integral part and
fractional part of the C̄,respectively**/

INITIALIZE  C̄1 = I
            C̄2 = I + 1

END


BEGIN  /*****  Use MVA to compute the
   performance measures for all feasible  states
   when  concurrency  is  approximated  by  the
   C̄1 and C̄2*****/

FOR  K=C̄1  TO  C̄2  DO
  : Compute  all  feasible  states  when  multi-
    programming level (MPL) is K.
   FOR  all state à  DO
     : Compute state transition matrix Q (à)
     : Compute state probability matrix p (à)
     : Use  MVA  for  computing  ρn (à),
       Ln (à) and λ0 (à)
   END  FOR
  : Compute ρn (K), Ln (K) and λ0 (K)
END  FOR
END
```

BEGIN ／＊＊＊＊Final computation＊＊＊＊／

$$\rho n(\bar{C}) = (1-F)\,\rho n(\bar{C}_1) + (F)\,\rho n(\bar{C}_2) \quad (10)$$

$$Ln(\bar{C}) = (1-F)\,Ln(\bar{C}_1) + (F)\,Ln(\bar{C}_2) \quad (11)$$

$$\lambda_0(\bar{C}) = (1-F)\,\lambda_0(\bar{C}_1) + (F)\,\lambda_0(\bar{C}_2) \quad (12)$$

END

図5．本手法のアルゴリズム

5．評価例

5．1　評価に用いるモデル

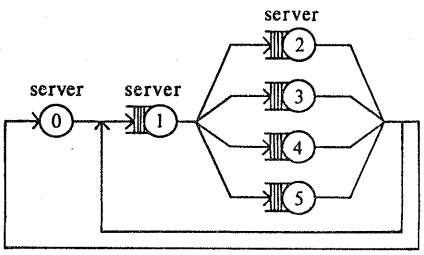

図6．評価用モデル

具体例として図6．に示す5つのＦＣＦＳサーバ（サーバ1～サーバ5）から成るネットワークを考える。サーバ1は平均サービス時間0．01秒、サーバ2～サーバ5は平均サービス時間0．04秒の指数分布に従う。サーバ間の結合確率を図7．に示す。

SERVER / SERVER	0	1	2	3	4	5
0	0	1	0	0	0	0
1	0	0	0.25	0.25	0.25	0.25
2	0.1	0.9	0	0	0	0
3	0.1	0.9	0	0	0	0
4	0.1	0.9	0	0	0	0
5	0.1	0.9	0	0	0	0

図7．サーバ間の結合確率

5．2　数値結果

以下のようなＤＣＰ，ＴＣＰ及びＶＣＰを図6．のモデルによって実行する場合について検討した。

ＤＣＰの場合については次のように4つの例を検討した。

case 1 ： N=2, C_1= {2} , C_2= {3}
case 2 ： N=3, C_1= {2} , C_2= {2} , C_3= {4}
case 3 ： N=4, C_1= {1} , C_2= {2} , C_3= {3} , C_4= {4}
case 4 ： N=5, C_1= {1} , C_2= {2} , C_3= {3} , C_4= {4} , C_5= {4}

ＴＣＰの場合については次のように4つの例を検討した。

case 1 ： N=2, C_1=C_2= {2,3}
case 2 ： N=3, C_1=C_2=C_3= {2,2,3}
case 3 ： N=4, C_1=C_2=C_3=C_4= {2,3,3,3}
case 4 ： N=5, C_1=C_2=C_3=C_4=C_5= {2,2,2,3,3}

ＶＣＰの場合については次のように4つの例を検討した。

case 1 ： N=2, C_1= {2,2,3} , C_2= {1,2,2}
case 2 ： N=3, C_1= {1,3,5} , C_2= {1,4,4} , C_3= {1,1,7}
case 3 ： N=4, C_1= {1,2,3,4} , C_2= {1,2,3} , C_3= {1,1,2} , C_4= {1,2,4}
case 4 ： N=5, C_1= {1,2,3,4,4} , C_2= {1,1,4} , C_3= {1,7} , C_4= {2,3,5} , C_5= {1,1,3}

本手法を用いた結果とシミュレーションによって求めた値を比較した。この結果の一部を表1．に示す。

ＶＣＰの場合は比較した結果によって一次タスク数を増やす（Ｎ＝5）と待ち行列長については最大約15％の誤差を生じることが分かった。これは一次タスク数が大きくなる時、並列度ベクトルＣｊの要素が大きくばらつくと近似精度が悪くなることが原因と考えられる。

また、本手法と分解近似の適用範囲を比較すると、本手法はＤＣＰ、ＴＣＰ、ＶＣＰのどの場合も適用できるが分解近似ではＴＣＰ及びＶＣＰの場合に対して適用不可能である（表2．参照）。

また、ＤＣＰの場合については分解近似を用いると状態数が非常に大きくなり、計算が困難である。一方平均並列度の概念を導入した本手法を用いると極めて少ない

状態数によって計算できるため、計算時間が短い、必要なメモリ量が少ない等の利点がある（表３．参照）。

現在この利点を利用してパソコン上にＣ言語を用いて解析ツールＡＴＬＡＳを開発している。

６．むすび

本研究ではＤＣＰ、ＴＣＰ、ＶＣＰの３つの変化する並列度を持つプログラム群を処理する並行処理システムを評価するため、並列度ベクトルを提案し、これに基づいた性能評価法を示した。本手法では非整数$\bar{C}$を$\bar{C}_1$と$\bar{C}_2$（＝$\bar{C}_1$＋１）に分解し、それぞれの場合において分解近似を用いて特性値を求める。そして$\bar{C}$の$\bar{C}_1$からの偏差を基いた比例配分によって$\bar{C}$の場合の特性値を求める。本手法の精度についてＤＣＰ、ＴＣＰ、ＶＣＰの様々な例によって本手法を用いて求めた値をシミュレーション結果と比較し、その有効性を示した。

本稿では、ＴＣＰとＶＣＰの場合には解析モデルのサイクルタイムを定数とし、近似を行った。今後は、サイクルタイムが異なる場合について検討する。また、二次タスクの同期処理に要するオーバヘッドを考慮した解析についても検討する予定である。

参考文献

(1) G.R.Andrews,F.B.Schneider : " Concepts and notations for concurrent programming ", ACM Computing Surveys,vol.15,No.1,pp.3-43(1983).

(2) A.Thomasian : " Performance evaluation of centralized databases with static locking", IEEE.Soft.Eng.,SE-11,pp.346-355(1985).

(3) A.Thomasian,P.F.Bay:"Analytic queueing network models for parallel processing of task systems", IEEE.Tran.Comp.,TC-35, pp. 1045-1054 (1986).

(4) K.M.Chandy,C.H.Sauer :"Approximate methods for analysing queueing network models of computer systems",ACM Computing Surveys,vol.10,No.3,pp. 281-317(1978).

(5) P.Heidelberger,K.S.Trivedi :"Analytic queueing models for programs with internal concurrency" ,IEEE.Tran.Comp.,TC-32,pp.73-82(1983).

(6) A.Bobbio,K.S.Trivedi:"An aggregation technique for the transient analysis of stiff Markov Chains",IEEE.Tran.Comp.,TC-35,pp.803-814(1986)

(7) D.Peng,K.G.Shin : "Modeling of concurrent task execution in a distributed system for real-time control",IEEE.Tran.Comp.,TC-36,pp.500-516 (1987).

(8) P.B.Hansen : " Distributed processes : A concurrent programming concept ", ACM Commu., vol.21,No.11,pp.934-941(1978).

(9) G.S.Graham : " Queueing network models of computer system performance",ACM Comp. Surveys ,vol.10,No.3,pp.219-224(1978).

(10)チャニントーン、渡辺、中西、真田、手塚 : " 並列性を持つプログラムを処理するシステムの性能評価" 、信学総全大、vol.8,pp.26 (1987).

(11)M.Reiser,S.S.Lavenberg : " Mean-value analysis of closed multichain queueing networks",ACM, vol.27,No.2,pp.313-322(1980).

(12)L.Kleinrock : "Queueing Systems,Vol. 1:Theory" ,John Wiley & Sons(1975).

性能評価	サーバ	DCP						TCP						VCP					
		CASE 1			CASE 2			CASE 1			CASE 2			CASE 1			CASE 2		
		SIM	APR	ERR	SIM	APR	ERR	SIM	APR	ERR	SIM	APR	ERR	SIM	APR	ERR	SIM	APR	ERR
待ち行列長	1	1.22	1.20	+1.6%	1.47	1.44	+2.0%	1.22	1.20	+1.6%	1.41	1.38	+2.1%	1.17	1.15	+1.7%	1.77	1.51	+14.6%
	2	1.19	1.20	-1.6%	1.39	1.44	-3.5%	1.18	1.20	-1.7%	1.36	1.38	-1.4%	1.15	1.15	+0.3%	1.60	1.51	+5.6%
	3	1.17	1.20	-2.5%	1.40	1.44	-2.8%	1.18	1.20	-1.7%	1.34	1.38	-2.9%	1.15	1.15	+0.3%	1.58	1.51	+4.4%
	4	1.19	1.20	-1.6%	1.40	1.44	-2.8%	1.20	1.20	+0.2%	1.36	1.38	-1.4%	1.14	1.15	-0.8%	1.60	1.51	+5.6%
	5	1.20	1.20	+0.4%	1.43	1.44	-0.6%	1.20	1.20	+0.3%	1.36	1.38	-1.4%	1.14	1.15	-0.9%	1.59	1.51	+5.2%
利用率	1	0.42	0.44	-4.7%	0.52	0.55	-5.7%	0.42	0.44	-4.7%	0.52	0.53	-1.9%	0.40	0.40	+1.1%	0.57	0.57	+1.0%
	2	0.43	0.44	-2.3%	0.52	0.55	-5.7%	0.42	0.44	-4.7%	0.51	0.53	-3.9%	0.39	0.40	-2.5%	0.57	0.57	+1.0%
	3	0.42	0.44	-4.7%	0.53	0.55	-3.7%	0.42	0.44	-4.7%	0.51	0.53	-3.9%	0.40	0.40	+1.0%	0.57	0.57	+1.0%
	4	0.42	0.44	-4.7%	0.52	0.55	-5.7%	0.42	0.44	-4.7%	0.52	0.53	-1.9%	0.40	0.40	+1.1%	0.57	0.57	+1.0%
	5	0.44	0.44	+0.5%	0.53	0.55	-3.7%	0.42	0.44	-4.7%	0.52	0.53	-1.9%	0.40	0.40	+1.2%	0.57	0.57	+1.0%
スループット (round/sec)	0	1.49	1.42	+4.6%	1.20	1.15	+4.1%	1.48	1.42	+4.0%	1.30	1.19	+8.4%	1.59	1.49	+6.2%	1.04	1.10	-5.7%

表1．結果の比較

プログラム群	分解近似	本手法
DCP	困難	可能
TCP	不可	可能
VCP	不可	可能

表2．本手法の適用範囲

プログラム群	状態数	
	分解近似	本手法
$C_1=\{1\}$ $C_2=\{2\}$ $C_3=\{3\}$	21	10
$C_1=\{2\}$ $C_2=\{3\}$ $C_3=\{4\}$	315	54
$C_1=\{1\}$ $C_2=\{1\}$ $C_3=\{3\}$ $C_4=\{3\}$	49	15
$C_1=\{2\}$ $C_2=\{2\}$ $C_3=\{4\}$ $C_4=\{4\}$	2025	105

表3．分解近似と本手法の比較（DCP）