RA-003

自由選択ネットの活性・安全性判定解析アルゴリズム改善と援用ツールへの実装

An Improved Algorithm and its Implement of the Liveness/Safeness Analysis for Free-Choice Net on Petri Net Tool

井出 和人 †　　**和崎 克己** ††
Kazuto Ide　Katsumi Wasaki

1　まえがき

離散事象システムにおけるモデル化手法の一つにペトリネット [1] がある．ペトリネットは視覚的で，かつ数学的なモデル化ツールである．筆者らの研究グループでは，ペトリネットの設計解析援用ツール HiPS[2] を開発している．HiPS には T-インバリアント解析などの解析機能が備わっているが，いくつかの重要な性質を解析する機能が不足している．本研究では，その重要な性質のひとつである，自由選択ネットのサブクラスにおける活性および安全性を判定するための並列実行可能なアルゴリズムの改善および実装を行った．

2　ペトリネット援用ツール HiPS

HiPS は，既存のペトリネットツールの記述性，操作性，再利用性の問題を解決するために本学で開発されたペトリネット援用ツールである [2]．図 1 に，HiPS の編集画面を示す．直感的で一般的な操作方法の GUI と，ペトリネットの階層化並びに時間ペトリネットに対応し，作成したペトリネットモデルをシミュレートし，挙動を観察することが可能である．また，T-インバリアント解析や可達解析などの解析機能が実装されており，現在も解析機能の追加が行われている．本ツールは著者らの研究グループによって，主に非同期回路設計・検証などに応用されている [3][4][5].

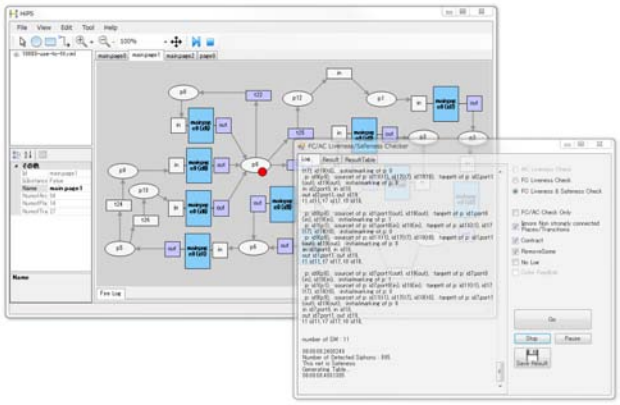


図 1　ペトリネット設計解析援用ツール HiPS の編集画面

† 信州大学大学院理工学系研究科,
Graduate School of Science and Technology, Shinshu University.
†† 信州大学工学部情報工学科,
Department of Computer Science and Engineering,
Faculty of Engineering, Shinshu University.

3　ペトリネットの活性・安全性判定

定義 1（活性）　活性とは，初期マーキング M_0 から達することができる全てのマーキング状態から，ネット内の任意のトランジションを何らかの経路で一回以上発火可能であるような性質である．ペトリネットが活性であるとき，モデル化されたシステムではデッドロックを生じない．

定義 2（安全性）　安全性とは，初期マーキング M_0 から達することができる全てのマーキング状態において，いずれのプレースにも 1 つを超えるトークンが入ることはないという性質である．ペトリネットが安全であるとき，モデル化されたシステムにおいてはリソースの有限な利用が保証される．

定義 3（自由選択ネット）　自由選択ネットとは，すべてのプレースからトランジションへのアークが，そのプレースからの唯一の出力アークであるか，トランジションへの唯一の入力アークであるようなペトリネットのことである．

定理 1（自由選択ネットの活性・安全性）　自由選択ネットは，ネット内のすべてのサイフォンがマーキングされたトラップを含むとき，かつその時に限り，活性である (文献 1:定理 12)．活性自由選択ネットは，M_0 においてそれぞれがただ 1 つのトークンを持つ強連結な SM-成分で覆われているとき，かつその時に限り，安全である (文献 1:定理 13).

4　活性判定アルゴリズムと改善点

定理 1 に基づいて，今回実装したアルゴリズムでは，(1) ネットの縮約 (判定効率化のため)，(2) 自由選択ネット判定，(3) サイフォンの列挙，(4) 列挙されたサイフォンにトラップが含まれているかの判定，(5) トラップ内にマーキングがあるかの判定，という手順で活性の判定を行う．

Step1 ネットの縮約【改善点 1】

今回実装したサイフォン生成アルゴリズムは，ネット内のすべてのプレースに対してサイフォンの生成を行う．このため，ネット内のプレース数が多ければ多いほど生成に時間がかかる．これを改善するために，縮約により，解析の必要なプレース数を減少させる．具体的には，$t_1{\bullet} = {\bullet}t_2 = \{p_1\}$($t_1{\bullet}$ は t_1 の出力プレース集合，${\bullet}t_2$ は t_2 の入力プレース集合) であるようなすべての t_1，t_2

および p_1 について，t_1 の出力プレースを t_2 の出力プレースに置き換える（＝ t_2 を入力に持つプレースの入力トランジション集合から t_2 を削除し t_1 を追加する）．その上で p_1，t_2 を削除する．

図 2 に縮約の実行例を示す．

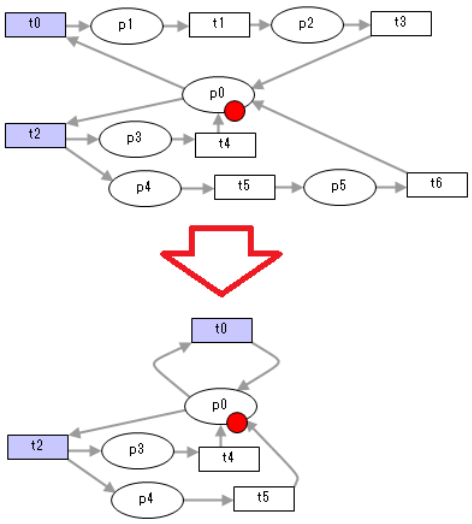


図 2　縮約機能の実行例

（実際には縮約は内部的に行われるため，
描画されているネット図が変化することはない）

Step2 自由選択ネットクラスの検査

作成されたペトリネットの構造が自由選択ネットの定義に合致するかどうかを確認する．ネット内のすべてのプレース p に対して $|p\cdot| = 1$ であるか，そうでなければ，p のすべての出力トランジション t に関して，$|\cdot t| = 1$ であるかを確認する ($|p\cdot|$ は p の出力トランジション数，$|\cdot t|$ は t の入力トランジション数)．合致しない場合はその部分を FC でないとして警告する．

Step3 サイフォンの列挙と重複削除【改善点 2】

サイフォンの列挙には既存研究 [6] に基づいたアルゴリズムを用いている．既存研究のアルゴリズムではネット内のすべてのプレースから始まり，順次プレースを追加していくことでサイフォンを生成する．これを一般に追加法と呼ぶ．ここで，追加するプレースは，初期プレースと追加されたプレースで作られるプレース集合に対する入力プレースである．

・同一プレース集合削除機能

追加法によりサイフォンの生成を行うと，同一のサイフォンが複数検出される場合がある．既存研究においては，サイフォンの生成作業をそれぞれのプレースに対して順次行っていくため，生成途中のプレース集合が既存のサイフォンと同一のものになるかどうかが判明しない．そのため，これら同一のサイフォンをサイフォン生成が完了した後に逐次除去するという方法をとっている．しかし，これは生成の必要がないサイフォンを生成していることになり，効率が悪い．そこで，今回実装したアルゴリズムでは，それぞれのプレースから開始するサイフォンの生成作業を，すべてのプレースに対して同時並列的に行う．これにより，生成途中において，最終的に同一のサイフォンとなる同一のプレース集合を知ることができる．これを利用し，生成途中のプレース集合の集合から同一のプレース集合を除去する機能を追加した．

この機能はマージソートに類似したアルゴリズムを用いている．各プレースは固有の ID を持つ．各プレース集合は含まれるプレースの ID の合計値を持つ．この合計値を比較し，一致すればさらに含まれるプレースの ID を比較し，同一のプレース集合かどうかを判定する．同一のプレース集合であれば生成途中のプレース集合の集合から一方を除去する．図 3 にこの機能の実行例を示す．

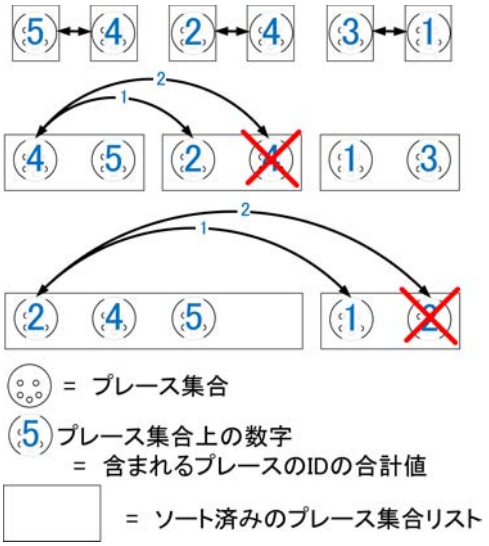


図 3　同一プレース集合削除機能の実行例

Algorithm1 に今回実装したサイフォン生成機能のアルゴリズムを示す．アルゴリズム中 (*) は今回追加した同一プレース集合削除機能の部分である．始めにネット内の個々のプレースを含むプレース数 1 個のプレース集合を作る．次にそれらのプレース集合がサイフォンであるかの判断をする．サイフォンでないプレース集合に対して入力プレースの追加を行う．その後も，サイフォンであるかの判断とプレースの追加を繰り返すことでサイフォンを生成していく．アルゴリズム中には記述がないが，実装においては.NET Framework 4.0 の Parallel.Foreach を利用し，各プレース集合に対してサイフォンであるかどうかの判断やプレースの追加を行う工程を並列実行し高速化を図っている．

Step4 トラップの検出

一般に削除法と呼ばれる手法を用いている．サイフォン内の他のプレースに対して，出力かつ入力でないプレースを削除していくことで，トラップを検出する．サイフォンにトラップが含まれていない場合は，全てのプレースが削除され，サイフォンからトラップを検出することはできない．並行化により，各サイフォンに対し同時に判定を行う．

Algorithm 1 サイフォン生成アルゴリズム

```
Require: 自由選択ネット
Ensure: ネット内のすべての基本サイフォンの検出
  for each Place in Net do
    List ⇐ Placeset ⇐ Place
  end for
  while List is not Empty do
    for All Placeset in List do
      if Placeset is Siphon then
        SiphonList ⇐ Placeset
      end if
      if Placeset has more SourcePlace then
        copyofPlaceset ⇐ SourcePlace
        List ⇐ copyofPlaceset
        remove Placeset from List
      end if
    end for
    Remove Same Placeset from List ——— (*)
  end while
  return SiphonList

  (”⇐” means ”Add”)
```

Step5 マーキングの検出

トラップのいずれかのプレースにマーキングが含まれているかどうかを検査する．

5 安全性判定アルゴリズムと改善点

定理 1 に基づいて，今回実装したアルゴリズムでは，(1) 強連結判定，(2) 活性判定 (上述 4 章で説明)，(3)P 閉部分ネットの列挙，(4) 列挙された P 閉部分ネットの SM-成分への分割試行，(5) ネットが SM-成分で覆われているかの判定，という手順で安全性の判定を行う．このうち活性判定は前述のものにより行う．

Step1 強連結性の検査

正規ペトリネットにおいて，活性でかつ強連結でない部分というのは，ソーストランジションかシンクプレースである．これらは安全でなく，また，強連結な SM-成分に分割することができない．このため，安全性判定の際にはペトリネットが強連結である必要がある．そこで，HiPS で作成されたペトリネットの構造が強連結であるかどうかを確認する．

この機能では強連結でない部分を取り除いたサブネットを得ることができる．これにより強連結でない部分を除外し，残りの部分について解析を行うといったことができる．

Step2 活性判定

４章において説明した手法で判定を行う．

Step3 P 閉部分ネットの検出

既存研究 [7](文献中の＜アルゴリズム 4 ＞) による．ネット内の各々のマークドプレースに対して以下の手順を行うことで P 閉部分ネットを導出する．

(1) ネットを複製する．
(2) 複製されたネットに対し以下の手順を実施する．
(2-1) 対象ののマークドプレース以外を削除する．
(2-2)2-1 によって生じた非強連結な部分を削除する．

実装においては，各マークドプレースに対する処理を並列化することで効率化を図っている．

Step4 SM-成分への分割と生成物の検査

既存研究 [7] に基づくアルゴリズムを用いている．同研究＜アルゴリズム 3 ＞について，非安全なネットにおいても SM-成分に分割可能であるように，また，より効率よく分割を行うために以下のように変更を行った．

1. 探索:マーキングを含むプレースから，分岐トランジション，合流トランジションで囲まれたパーツを取得し，生成中 SM に追加する．
2. 選択:1 つの分岐トランジションについて，出力プレースの 1 つを選ぶ．その他のプレースとそのプレースを含むパーツを選択不可能としてマークする．その他のプレースの数に応じて生成中 SM のコピーを作成，保存する．（この際選択を行う分岐トランジションは，選択の必要な分岐トランジションのうち，最も新しい探索手順によって追加されたトランジションである．）
3. 探索:選択された出力プレースを含むパーツを取得し，生成中 SM に追加する．
4. さらに分岐トランジションがある場合は 2 を実行する．既に選択済みの部分に到達し，探索も選択も行う必要がなければ，１つの SM-成分の完成である．選択不可としてマークされた部分により選択作業が続行できない場合は，対象の P 閉部分ネットは SM-成分に分割不可であるので分割作業を終了する．
5. コピーされた生成中 SM に関して 2 を実行する．その際に，すでに完成した SM-成分で使用されたプレースは選択肢から除外する．
6. すべての生成中 SM が，SM-成分となるか，選択も探索もできない状態になったら SM 分割を完了とする．

また，既存研究 [7] のアルゴリズムは活性安全自由選択ネットに対するものであることから，図 4 に示すような SM でない生成物が生成される場合がある．そこで，生成物がステートマシンであるか判定し，SM でないものを除去する機能を追加した．具体的には，生成物のすべてのトランジションの入力が 1 つであるか，出力が一つであるかを検査し，この条件を満たさない場合は SM でないので除去する．

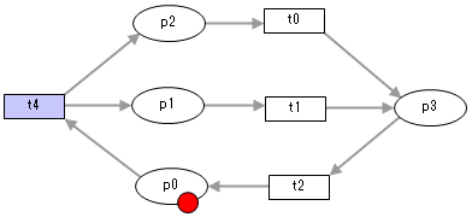


図 4　SM でない生成物の例

Algorithm2 に今回実装した SM-成分への分割アルゴリズムを示す．実装に際しては，各 P 閉部分ネットに対する分割作業を並列実行し，効率化を図っている．

Algorithm 2 State-Machine 分割アルゴリズム

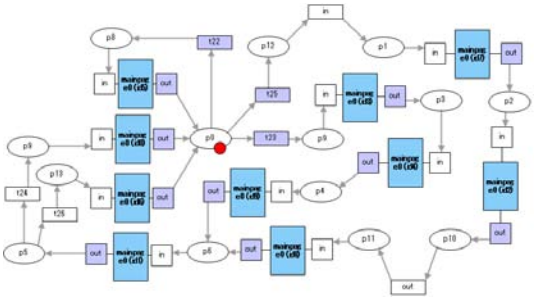
```
Require: P 閉部分ネット
Ensure: ネットの SM-成分への分割
  List ⇐ GeneratingSM ⇐ MarkedPlace
  GeneratingSM ⇐ PartOfNet(MarkedPlace)
  while List is not Empty do
    GeneratingSM = List[0]
    while blanch Transition exist in GeneratingSM do
      for each branch Transitions do
        BeforeSelect = copy of GeneratingSM
        select one Place from unused selectable Target Places
        for each unselected unused selectable Target Places do
          part of Net = PartOfNet(one of Target Place)
          mark part of Net as unselectable
        end for
        GeneratingSM ⇐ PartOfNet(selected Place)
        if number of unselected branch > 0 then
          List ⇐ copy of BeforeSelect
        end if
      end for
    end while
    GeneratedSMs ⇐ GeneratingSM
    Remove List[0]
  end while
  for each GeneratedSM do
    check is SM
    if GeneratedSM is not SM then
      Remove from GeneratedSMs
    end if
  end for

  (PartOfNet(Place) means part of Net which connected to Place, without Target of branch Transitions, without Source of merge Transitions)(”⇐” means ”Add”)
```

Step5 被覆検査

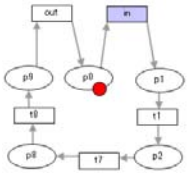
ネットが SM-成分で覆われているかを判定する．具体的には，生成された SM-成分に含まれる要素を覆われているとしてマークする．これにより，ネット内のすべての要素がマークされれば安全，そうでなければ非安全となる．

6 サイフォン生成数が大きいケースにおける活性・安全性判定とアルゴリズム改善の効果

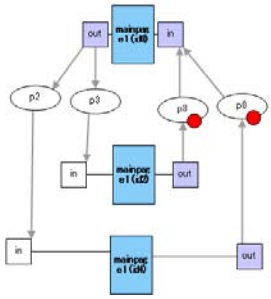
図 5(a)(c)(d) のような，3 種類のペトリネットについて活性・安全性判定を行った．結果としては，(a) は 1,457 個 のサイフォンと 11 個のステートマシン，(c) は 72,122 個のサイフォンと 35 個のステートマシン，(d) は 642,496 個のサイフォンと 93 個のステートマシン，がそれぞれ検出され，全てのサイフォンがマーキングされたトラップを含み，また，いづれのネットもステートマシンで覆うことができたため，定理 1 を満たしており活性・安全性を有することが判定できた．



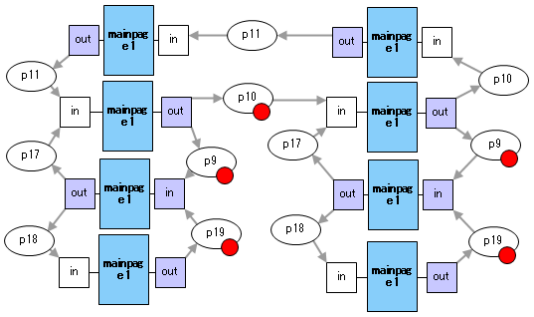
(a)(c) および (d) 内で使用されているサブページ ($|p| = 64, |t| = 57$)



(b)(a) 内で使用されているサブページ



(c) ペトリネットモデル例 1 ($|p| = 196, |t| = 171$)



(d) ペトリネットモデル例 2 ($|p| = 524, |t| = 456$)

図 5　判定テストに使用したペトリネット

これらの判定解析について，改善点 1 および改善点 2 の使用・不使用によって解析時間やメモリ使用量にどれだけの差が出るかを調査した．実装と性能評価に使用した環境は表 1 の通りである．

図 6 にそれぞれのペトリネットの判定にかかった時間および使用されたメモリ容量に関するグラフを示す．

時間に関しては，いずれの場合も双方の改善点を使用した場合が最も早く，(a) においては 523msec が

表 1　実行環境

CPU	Core i7 970 3.2GHz
Memory	DDR3 1333 18GB
OS	Windows7 Professional 64bit
実行バイナリ生成	Visual Studio 2010 Ultimate C#

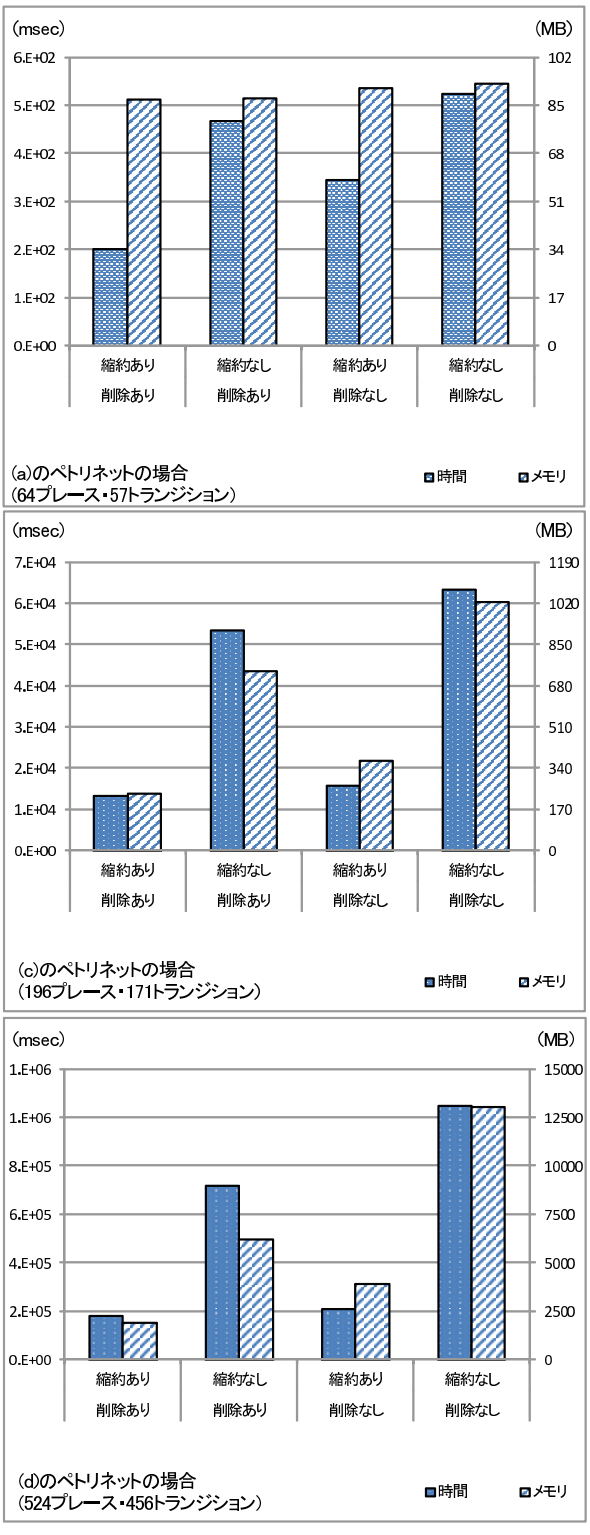


図 6　各ペトリネットの判定にかかった時間およびメモリ容量

201msec になり，双方の機能を使わない場合のおよそ 2 分の 1，(c) においては 63.2sec が 13.1sec に，(d) においては 1050sec が 178sec になり，双方の機能を使わない場合のおよそ 5 分の 1 の時間で判定を完了することができた．

メモリ使用量に関しては，(a) のネットではほとんど差が出なかったが（これは，判定のために使用するメモリ容量が十分に小さく，ペトリネットの構造情報を保持するために使われるメモリ容量に隠れてしまうためと考えられる），(c) においては 1020MB が 233MB になりおよそ 4 分の 1 に，(d) においては 13.1GB が 1.92GB になり，7 分の 1 近くまでメモリ使用量を低減することができた．

これらの結果から，縮約機能・同一プレース集合削除双方とも時間短縮，メモリ節約に貢献しており，双方の改善は有意に機能していることが明らかになった．

7　まとめ

今回のアルゴリズム改善によって，既存研究 [6] のアルゴリズムよりいくつかの手順を省略することができた．サイフォン生成数が多く，大規模なネットであれば，この改善によりさらなる効率化が期待できる．

今回追加した機能によって援用ツール上でペトリネットの活性・安全性判定およびネットの SM-成分への分割が行えるようになった．

今後の課題として，非対称選択ネットでの活性判定機能の実現，活性安全自由選択ネットのマークグラフへの分割機能，ならびに，CUBLAS+GPGPU アクセラレータを用いた解析器のさらなる高速化などが挙げられる．

謝辞 本研究の一部は科学研究費 (23500174) の助成を受けたものである．

参考文献

[1] Tadao Murata：Petri Net: Propaties, Analysis and Applications, IEEE, Vol.77, No.4, pp. 541-580, 1989.
[2] https://sourceforge.net/projects/hips-tools/
[3] 松山千尋, 和崎克己：Time-Petri Net を用いた非同期回路のモデル化と階層化設計；FIT2009（第 8 回情報科学技術フォーラム）講演論文集, pp. 523-526, 2009.
[4] 關屋貴詞, 和崎克己：並列制御システム設計の UML アクティビティ図に対するペトリネットによる正当性検証；FIT2011（第 10 回情報科学技術フォーラム）講演論文集, pp. 253-256, 2011.
[5] 堀内維作, 和崎克己：高水準ペトリネットを記述可能な援用ツール HiPS2 と非同期回路検証への適用；FIT2011（第 10 回情報科学技術フォーラム）講演論文集, pp. 393-396, 2011.
[6] Erwin R. Boer, et al.：IEEE Trans. Circ. and Syst-I, Vol.41, No.4, pp. 266-271, 1994.
[7] 西村忠昭，他：ライブかつセーフな自由選択ネットの分割アルゴリズム；電子情報通信学会論文誌 A Vol.J76-A No.11, pp. 1547-1556, 1993.