縮小探索行列法による不完全指定順序回路の最大両立性クラスの生成法

後藤公雄・巽　久行

A Method of Generation of Maximal Compatible Classes in the Incompletely Specified Sequential Machines by Using Reduced Search and Matrix Algorithm

Kimio GOTO and Hisayuki TATSUMI

Abstract

In the methods of generating the Maximum Compatible Classes in the Incompletely Specified Sequential Machine, the Stoffers' method has been well known as one of the methods dependent on the Compatible Pairs, that is, the CP dependent methods. But the purpose of the method of this paper is to decrease the computer operating time taken to generate the MCC by using the Stoffers' method. The method in this paper tries to search less combinations of internal states than the Stoffers' method. In the method of this paper, the matrix of the Compatible Class is used, and a row product set between some several row sets in that matrix is compared with the sum set of the alphabet numbers of all rows which have connected with the generation of this row product set. As a result of such comparison it is decided whether this row product set should be grown up to the higher row product sets or it should be shrunk to the lower row product sets. For this purpose, the three main theorems are derived as the standards of such decisions. In addition, this method was compared with the Stoffers' method by running the BASIC program of each method, in terms of the computer operating times. Then it is confirmed that the operating time by the method of this paper is from 1/2 to 1/4 shorter than that of the Stoffers' method.

1. まえがき

不完全指定順序回路の最小化[1,2,7~9,11]後の新内部状態の構成源となる最大両立性クラス(Maximum Compatible Class, 以後MCCと呼ぶ)の生成法として，古くから大別して2つの方法が知られている。その一つはインプリケーション・テーブルから求まった非両立性対（Incompatible Pair, 以後ICPと呼ぶ）を基にするものであり，他の一つは同じテーブルから求まる両立性対（Compatible Pair, 以後CPと呼ぶ）を基にするものである。前者は与えられたすべての内部状態の集合を，同一の集合の中に同じICPの要素を同時に含むことがないように逐次分割してゆく方法である[1,3~5,10,12]。また，後者は，どの2つの構成要素を組み合わせてもCPとなり，しかも集合にはカバーされない最大の集合を生成する方法である。この方法にはUnger[1]やStoffers[6,13]の方法が提案されている。特にStoffersの方法では，両立性マトリックスと呼ばれる行列上に記入されたCPを行ごとに組み合わせてMCCを生成する。この手法では，次にどの組合わせを選んでMCC生成を行うべきかについて，その都度調査するようにしており，これによってMCC生成のための探索範囲を限定し，縮小している。

本論文では，まず，CPの記入により作られるCC行列，その各行要素間で作られる行線積（集合）と起原集合および全1行列などの考えから出発する。さらにこれらの2種類の集合相互間の文字要素の有無によっ

平成3年9月5日受理

* 情報工学科

てMCCの探索方向として前進をとるか，後退をとるかを判定する3つの主要定理を導入している。これによってMCCの探索範囲がStoffersの手法よりもさらに狭く限定でき，MCCの算出時間をさらに短縮することができる。なお，実際に本論文による手法とStoffersの手法によるプログラムを計算機上で実行させて，特に演算時間の面からこの手法の有効性を検証する。

このように，本論文で提案する方法は縮小探索の手法をさらに改善する方法である。

2. 本論文の手法の定義と定理

本章では，本論文の手法によってMCCを生成するための定義と定理を述べる。

［定義1］ 不完全指定順序回路の内部状態を x_i（ここで，$1 \leq i \leq n$）とし，これらのリテラルに

$$x_i \leq x_{i+1} \tag{1}$$

のような順序づけをして表現したとき，x_{i+1} は x_i よりも順位が高いと呼ぶ。このように順位の高いリテラル程右，または下に来るように並べたとき，この配列を昇順配列と呼ぶ。また，逆に順位の低いリテラル程右，または下に来るように並べた配列を降順配列と呼ぶ。

［定義2］ 不完全指定順序回路の内部状態 x_i に対応する行（列）線を行（列）x_i と呼ぶ。

［定義3］ 不完全指定順序回路の内部状態 x_i と x_j から作られる両立性対 x_ix_j があるとき，行 x_i と列 x_i，行 x_i と列 x_j，行 x_j と列 x_i，および行 x_j と列 x_j のそれぞれの交点のすべてに1を記入する。すべてのCPについてこれを行って生成される行列を両立性クラス行列（Compatible Class Matrix，以後略してCC行列）と呼ぶ。ただし，各行と各列はすべて昇順に配列されているものとする。

［定義4］ CC行列の行 x_r 上で列 x_c との交点に1が存在すれば，行 x_r は要素 x_c を持つと呼ぶ。また行 x_i が要素 $x_1, x_2, \cdots, x_k$ を持つとき，この要素の集合 $\boldsymbol{r}(x_i)$ を行 x_i の要素集合と呼び，次のように書く。

$$\boldsymbol{r}(x_i)=\{x_1, x_2, \cdots, x_k\}. \tag{2}$$

［例1］ $a \leq b \leq c \leq d \leq e \leq f$ の順序関係を持つ6個の内部状態は *abcdef* のように昇順配列される（定義1）。このような6個の内部状態に対応する行線 *a*, *b*, …, *f* と列線 *a*, *b*, …, *f* によって図1の行列が得られる（定義2）。不完全指定順序回路のCPとして与えられた *ab*, *ac*, *ad*, *ae*, *af*, *bc*, *bd*, *ce*, *cf*, *de*, *df* および *ef* を，定義3で述べたように各行線と各列線の交点に1として記入すると，図1自身がCC行列となる。この図で各行線はいくつかの要素を持つ。たとえば行線 *a* は要素 *a*, *b*, *c*, …, *f* を持ち，行線 *a* の要素集合は次式となる（定義4）。

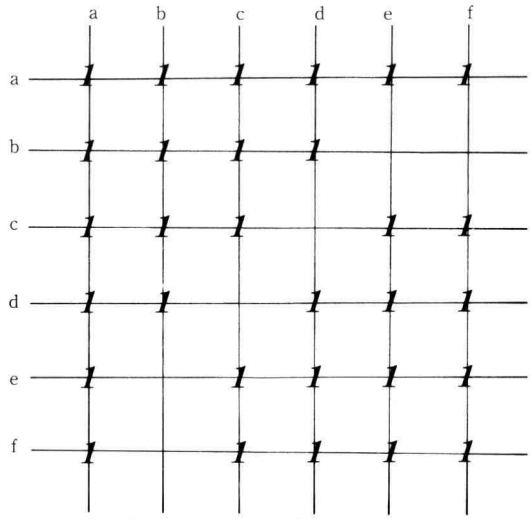


図1. CPが *ab*, *ac*, *ad*, *ae*, *af*, *bc*, *bd*, *ce*, *cf*, *de*, *df* および *ef* の場合のCC行列

$$\boldsymbol{r}(a)=\{a, b, c, d, e, f\}. \tag{3}$$

［定義5］ CC行列で，行 $x_1, x_2, \cdots, x_j, \cdots, x_n$ 各々の要素集合の積集合を $\boldsymbol{r}(x_1, x_2, \cdots, x_j, \cdots, x_n)$ と書くと，

$$\boldsymbol{r}(x_1, x_2, \cdots, x_j, \cdots, x_n)=\prod_{j=1}^{n} \boldsymbol{r}(x_j) \tag{4}$$

が成立する。この $\boldsymbol{r}(x_1, x_2, \cdots, x_j, \cdots, x_n)$ を行 $x_1, x_2, \cdots, x_j, \cdots, x_n$ の行線積と呼び，この生起源となる各行 $x_1, x_2, \cdots, x_j, \cdots, x_n$ 自体の集合をこの行線積の起源集合と呼ぶ。

［定義6］ CC行列で行 $x_1, x_2, \cdots, x_k, x_{k+1}, x_{k+2}, \cdots, x_{k+l}$ の行線積 $\boldsymbol{r}(x_1, x_2, \cdots, x_k, x_{k+1}, x_{k+2}, \cdots, x_{k+l})$ は行 $x_1, x_2, \cdots, x_k$ の行線積 $\boldsymbol{r}(x_1, x_2, \cdots, x_k)$ よりも l 次だけ高い行線積であると呼ぶ。特に $x_k \leq x_{k+1} \leq x_{k+2} \leq \cdots \leq x_{k+l}$ のとき，この行線積を l 次高次の昇順行線積と呼ぶ。

［例2］ 図1で行 *b* の要素集合 $\boldsymbol{r}(b)$ は，

$$\boldsymbol{r}(b)=\{a, b, c, d\} \tag{5}$$

となる。式(3), (5)より行線積 $\boldsymbol{r}(a, b)$ は,

$$\boldsymbol{r}(a, b)=\boldsymbol{r}(a)\cap\boldsymbol{r}(b)=\{a, b, c, d\} \tag{6}$$

となり，この行線積の起源集合は $\{a, b\}$ となる（定義5)。また，行線積 $\boldsymbol{r}(a, b, c, d)$ は行線積 $\boldsymbol{r}(a, b)$ よりも2次高い昇順行線積 $\{a, b\}$ を与える（定義6)。

［定義7］ CC行列で複数個の行があり，さらにそれらの行と名称が全く一致する列があるとき，これらの行と列の全交点上に1が存在するものを全1行列と呼ぶ。

［定義8］ CC行列の全1行列の中で他の全1行列には包含されない最大の（行数，列数の最も多い）行列をMCC行列と呼ぶ。

［例3］ 図1で，行と列がともに a と b より成る行列と，ともに a, b および c より成る行列は，すべての交点に1を持つので全1行列である。しかし前者は後者に包含され，後者は他の全1行列には包含されないので，後者のみがMCC行列になる。

［定理1］ CC行列で行と列に同じ要素 $x_1, x_2, \cdots, x_k$ を用いて k 行 k 列以上の全1行列を作り得るための必要十分条件は，行線積 $\boldsymbol{r}(x_1, x_2, \cdots, x_k)$ がその起原集合 $\{x_1, x_2, \cdots, x_k\}$ をカバーすることである。すなわち，

$$\boldsymbol{r}(x_1, x_2, \cdots, x_k)\supseteq\{x_1, x_2, \cdots, x_k\} \tag{7}$$

が成立することである。

（証明） かりに式(7)が成立しないものとすると，

$$\boldsymbol{r}(x_1, x_2, \cdots, x_k)\subset\{x_1, x_2, \cdots, x_k\}$$

が成立するから，$1\leq i\leq k$ の範囲のすべての i について要素集合 $\boldsymbol{r}(x_i)$ は，たとえば最大 $k-1$ 個の要素 $x_1, x_2, \cdots, x_k$ のみを共通に持ち，それ以外の要素は決して共通には持ち得ない。したがって k 個の要素 $x_1, x_2, \cdots, x_k$ を用いても $k-1$ 行 $k-1$ 列以下の行1行列しか作り得ない。したがって題意を満足するには式(7)が成立せねばならない。この逆も容易に証明できる。
（QED）

［定理2］ CC行列で行と列が共に $x_1, x_2, \cdots, x_k$ より作られる全1行列が，k 行 k 列のMCC行列となるための必要十分条件は，行線積 $\boldsymbol{r}(x_1, x_2, \cdots, x_k)$ はその起原集合 $\{x_1, x_2, \cdots, x_k\}$ そのものとなることである。すなわち，

$$\boldsymbol{r}(x_1, x_2, \cdots, x_k)=\{x_1, x_2, \cdots, x_k\} \tag{8}$$

が成立することである。

（証明） 式(8)が成立しないものとすると，次の式のどちらか一方が成立せねばならない。

$$\boldsymbol{r}(x_1, x_2, \cdots, x_k)\subset\{x_1, x_2, \cdots, x_k\}, \tag{9a}$$
$$\boldsymbol{r}(x_1, x_2, \cdots, x_k)\supset\{x_1, x_2, \cdots, x_k\}. \tag{9b}$$

式(9a)の成立するときは，定理1の証明と同様に k 行 k 列より小さな行列しか得られない。また，式(9b)が成立するときは，行線積 $\boldsymbol{r}(x_1, x_2, \cdots, x_k)$ は，より高い順位の列要素 x_u を含む。すなわち，

$$\boldsymbol{r}(x_1, x_2, \cdots, x_k)\supseteq\{x_1, x_2, \cdots, x_k, x_u\}. \tag{10}$$

また，式(10)が成立するときは，CC行列の対称行列としての性質より，行 x_u の要素集合 $\boldsymbol{r}(x_u)$ は，

$$\boldsymbol{r}(x_u)\supseteq\{x_1, x_2, \cdots, x_k, x_u\}. \tag{11}$$

式(10)と(11)より，

$$\boldsymbol{r}(x_1, x_2, \cdots, x_k, x_u)\supseteq\{x_1, x_2, \cdots, x_k, x_u\}. \tag{12}$$

これより $x_1, x_2, \cdots, x_k, x_u$ より成る行列は $k+1$ 行 $k+1$ 列以上の全1行列を作れることになり（定理1)，$x_1, x_2, \cdots, x_k$ よりなる行列は最大の全1行列，すなわちMCCとはなり得ない（定義8)。これより題意を満足するには式(8)が成立せねばならない。この逆も容易に証明できる。
（QED）

［説明1］ 図1で行と列が共に a と b の行列は2行2列の全1行列で，定理1の条件 $\boldsymbol{r}(a, b)\supseteq\{a, b\}$ を満足する。この行列は，同じ条件によりさらに行と列が a, b および c より成る3行3列の全1行列に成長させることができる。特に後者の行列は定理2の条件 $\boldsymbol{r}(a, b, c)=\{a, b, c\}$ を満足し，MCC行列となる。

［定理3］ CC行列で，ある行線積の起原集合がその行線積に含まれない要素を持てば，この行線積およびその高次行線積にたいする起原集合の要素で全1行列を作ることはできない。

（証明） 題意より，q 個の行 $x_1, x_2, \cdots, x_{k-1}, x_k, x_{k+1}, \cdots, x_q$ にたいする行線積が，その要素として x_k を含まないものとすると，この行線積の高次行線積を作っても，その要素として x_k は含まれない。したがって最初の行線積とそれより得られる高次行線積はいかに高次になっても起原集合をカバーすることはあり得ない。したがって定理1が成立せず，q 個の行および列の要素で q 行 q 列以上の全1行列を作ることができない。
（QED）

［説明2］ 図1で行 a, b および e の行線積 $\boldsymbol{M}_1$ は,

$$\boldsymbol{M}_1=\boldsymbol{r}(a,\ b,\ e)=\{a,\ c,\ d\}$$

となり，その起原集合 $\{a, b, e\}$ は $\boldsymbol{M}_1$ に含まれない要素 b と e を持つ。また $\boldsymbol{M}_1$ よりも1次高い行線積 $\boldsymbol{M}_2$ は，

$$\boldsymbol{M}_2=\boldsymbol{r}(a,\ b,\ e,\ f)=\{a,\ c,\ d\}$$

となる。$\boldsymbol{M}_1$ と $\boldsymbol{M}_2$ の起原集合の要素で全1行列が得られないことは明らかである。

［定理4］ CC行列で，ある行にたいする行線積がその起原集合に含まれない要素を持つとき，これらの要素の中に起原集合の最高位の要素 x_k よりも高位のものが少なくとも1個あれば，この行線積を成長させ作られる昇順高次行線積の起原集合の要素で全1行列を作り得る。また，このように x_k よりも高位のものがなければ，元の行線積を成長させて得られる昇順高次行線積の起原集合から全1行列を作ることはできない。

（証明） いま，行 $x_1, x_2, \cdots, x_i, x_k$ および x_l の行線積が次のように表されるものとする。

$$\boldsymbol{r}(x_1,\ x_2,\ \cdots,\ x_i,\ x_k,\ x_l)=\{x_1,\ x_2,\ \cdots,\ x_i,\ x_j,\ x_k,\ x_l,\ \delta\cdot x_{l+m}\}. \tag{13}$$

ここで，

$$\delta=1 \quad \text{または}\ 0, \tag{14}$$

$$x_1\le x_2\le\cdots\le x_i\le x_j\le x_k\le x_l\le x_{l+m}. \tag{15}$$

$\delta=1$ のときは，式(13)の行線積の要素の中で，x_{l+m} は起原集合の最高位の要素 x_l よりも高位であり，また x_j は x_l よりも低位であることが式(15)より明らかである。しかもこれらの2つの要素は式(13)の行線積の起原集合には含まれていない。このとき式(13)とCC行列の対称行列としての性質から，

$$\boldsymbol{r}(x_{l+m})\supseteq\{x_1,\ x_2,\ \cdots,\ x_i,\ x_k,\ x_l,\ x_{l+m}\}. \tag{16}$$

式(13)と(16)から，

$$\boldsymbol{r}(x_1,\ x_2,\ \cdots,\ x_i,\ x_k,\ x_l,\ x_{l+m})\supseteq\{x_1,\ x_2,\ \cdots,\ x_i,\ x_k,\ x_l,\ x_{l+m}\}. \tag{17}$$

式(17)より $x_1, x_2, \cdots, x_i, x_k, x_l$ および x_{l+m} で表される行と列によって全1行列が得られることが定理1によって明らかとなる。

$\delta=0$ のときは，起原集合の最高位要素 x_l よりも低位の要素 x_j が式(13)の行線積に含まれる。この場合には行線積の最高位要素が x_l であるから，x_l よりも高位の要素を持ってきて式(13)より高次の行線積を作っても，その集合の最高位要素は x_l を越えることはあり得ない。したがってこのような高次行線積の起原集合の要素で全1行列は作り得ない。　（QED）

［説明3］ 図1で次に示す行線積が得られる。

$$\boldsymbol{r}(a,\ d)=\{a,\ b,\ d,\ e,\ f\}, \tag{18a}$$

$$\boldsymbol{r}(a,\ d,\ e)=\{a,\ d,\ e,\ f\}, \tag{18b}$$

$$\boldsymbol{r}(b,\ c)=\{a,\ b,\ c\}. \tag{18c}$$

式(18a)の行線積の要素には，その起原集合の最高位の要素 d よりも高位の要素 e と f が，また d よりも低位の要素 b が含まれている。このとき式(18b)に示すより高次の行線積 $\boldsymbol{r}(a, d, e)$ を求めると，式から明らかなようにこの行線積はその起原集合をカバーしているから，要素 a, b および e によって全1行列を作り得る。この例は定理4の前半に相当する。

式(18c)の行線積には起原集合の最高位の要素 c よりも高位の要素は含まれておらず，しかも起原集合に含まれない行線積の要素は c より低位の a のみである。このような場合，式(18c)よりも高次の行線積を作っても，その起原集合の要素から全1行列を作ることはできない。これは定理4の後半に相当する。

3. アルゴリズムと計算例

本章では前章で述べた定義と定理に従って本論文の手法によるMCC生成のアルゴリズムを述べ，さらにその計算例を示す。

3.1 本論文の手法によるアルゴリズム

つぎの各ステップに従ってCC行列を用いてMCCを生成する。ここで便宜上，行線積とその起原集合を列要素の文字列として表し，しかも計算の繰返し過程を明らかにするため，これらをそれぞれ配列 $mcp^{(no)}$ と $mcs^{(no)}$ に格納することとする。なお，このアルゴリズムのフローチャートを図2に示す。

［ステップ1］ 計算のための諸元および大きさの設定を行う。

［ステップ2］ 内部状態数 n，CPの個数numcpおよびCPをすべて入力する。

［ステップ3］ CC行列のすべての k 行について文字列 $CC(k)$ を生成する。

［ステップ4］ 最低位から昇順に変化する内部状態番号 i を更新・設定する。i の順位を判定し，最高位より1つ下の位の文字を終了すれば終る。

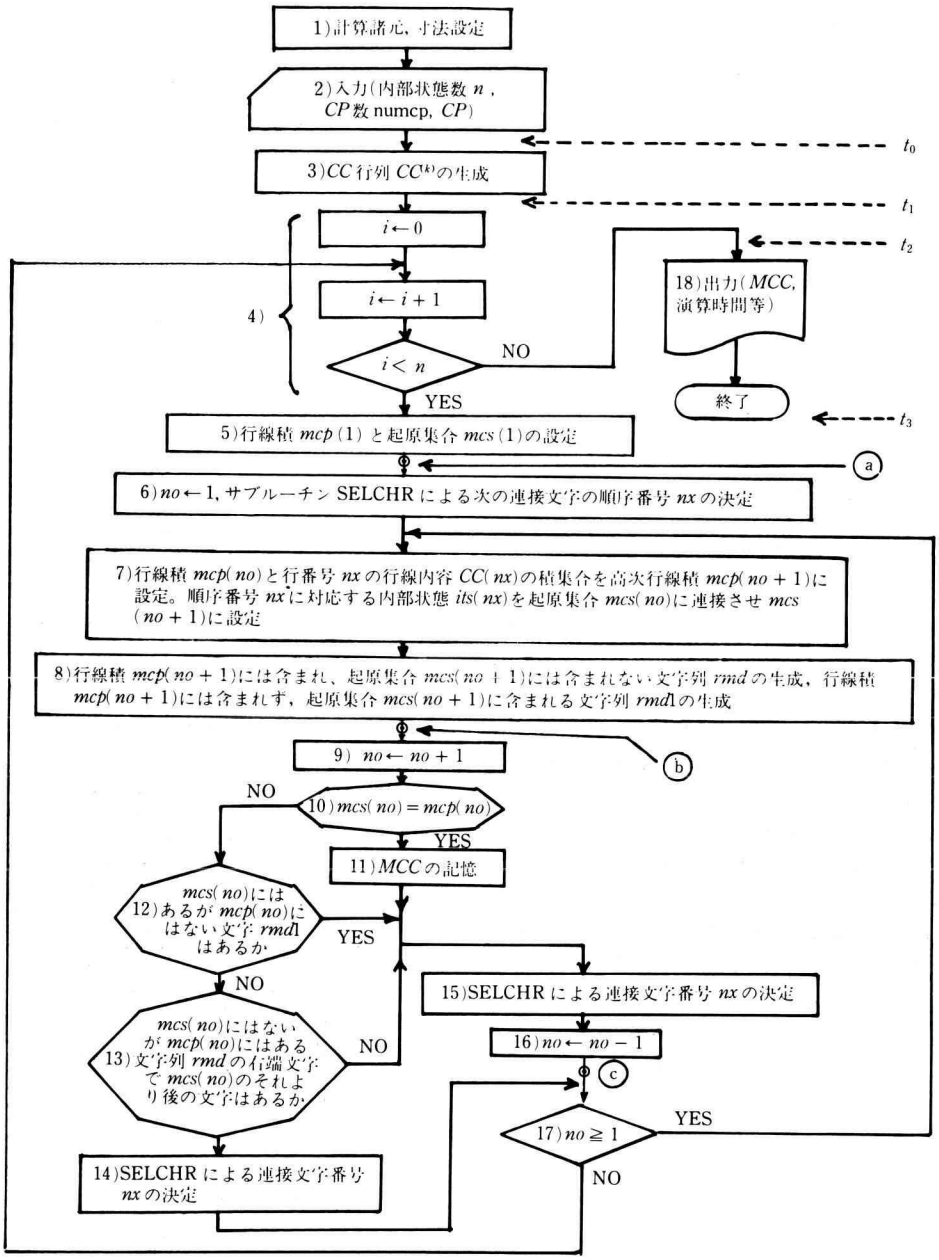


図2．本論文の手法のアルゴリズムのフローチャート

［ステップ5］ 内部状態 i に対する対応する CC 行列の行 i の文字列 $CC^{(i)}$ を行線積 $mcp^{(1)}$ に初期設定し，この行線積の起原集合 $mcs^{(1)}$ に文字 i を初期設定する。

［ステップ6］ 行線積更新回数 no に1を設定し，連接文字決定をサブルーチン（以後 SELCHR と略称する）を用いて連接される文字の順序番号 nx を決定する。

［ステップ7］　nx 番目の行で1の存在する列に対応する文字より成る文字列 $CC^{(nx)}$ と行線積 $mcp^{(no)}$ との積集合を求め，これを1次だけ高い行線積 $mcp^{(no+1)}$ とする。さらに起原集合 $mcs^{(no)}$ の右に nx に対応する文字 $its(nx)$ を連接させ，高次行線積 $mcp^{(no+1)}$ の起原集合 $mcs^{(no+1)}$ を生成する。

［ステップ8］　高次行線積 $mcp^{(no+1)}$ に含まれるすべての文字を抽出し，それらがそれぞれ高次起原集合 $mcs^{(no+1)}$ に含まれるか否か確認し，含まれないものがあれば，これらを文字列として rmd に格納する。また高次起原集合 $mcs^{(no+1)}$ に含まれるすべての文字を抽出し，それらがそれぞれ高次行線積 $mcp^{(no+1)}$ に含まれるか否か確認し，含まれないものがあれば，これらを文字列として $rmd1$ に格納する。

［ステップ9］　行線積更新回数 $no+1$ を no に書き直す（これは no の1インクリメントではない）。

［ステップ10］　ステップ8で求めた rmd が"空"でないか，または $rmd1$ が"空"でなければ，すなわち rmd と $rmd1$ の少なくとも一方が"空"でない場合にはステップ12に移る。そうでない場合，すなわち rmd と $rmd1$ の両方が"空"の場合は，行線積と起原集合の内容の文字列が完全に一致している場合であり，定理2よりこの場合の行線積はMCCである。この場合はステップ11へ移る。

［ステップ11］　文字列 $mcp^{(no)}$ を $mcc^{(nomcc)}$ にMCCとして記憶させ，ステップ15に移る。ここで $nomcc$ はMCCの生起順位を表す。

［ステップ12］　$rmd1$ が"空"でないか否か判定する。すなわち起原集合 $mcs^{(no)}$ には存在するが，行線積 $mcp^{(no)}$ には存在しない文字があるか否か判定する。このような文字があれば，すなわち $rmd1$ が"空"でなければ，定理3よりさらに高次の行線積は用いて全1行列を作ることはできないので，ステップ15に移る。そうでなければ，すなわち $rmd1$ が"空"であれば，ステップ13に移る。

［ステップ13］　ここで，もし rmd が"空"であれば，ステップ10でNOと判定される場合と同じであるので，このステップでは rmd が"空"でない場合，すなわち rmd に文字が存在する場合について調べる。このため，行線積 $mcp^{(no)}$ の起原集合 $mcs^{(no)}$ には存在しないが，$mcp^{(no)}$ には存在する文字がアルファベット順に配列された文字列 rmd 中の右端文字で，$mcs^{(no)}$ の右端文字よりもアルファベットの順番が後のものが少なくとも1個存在するか否か調べる。このためには，rmd の右端文字が $mcs^{(no)}$ の右端文字よりもアルファベットの順番が後になっているか否か調べればよい。もし後になっているなら，定理4によりさらに高次の行線積を生成し得るので，ステップ14に行く。もしそうでなければ，このままでは高次の行線積は実現できず，ステップ15に行く。

［ステップ14］　サブルーチンSELCHRを呼び，現在の行線積更新回数 no，起原集合 $mcs^{(no)}$ およびCC行列の行 i の要素内容を表す文字列 $cc^{(i)}$ より，次に連接する内部状態文字の番号 nx を決定する。

［ステップ15］　ステップ14と同じことを行う。

［ステップ16］　行線積更新回数 no を1だけデクリメントする。

［ステップ17］　行線積更新回数 no が1以上であればステップ7に戻ってそれ以降を繰り返し，そうでなければステップ4に戻って新たな i を設定する。

なお，すでに述べたサブルーチンSELCHRについて述べる。このサブルーチンは，MCC生成のための行線積の組合せの探索が前進ステップまたは後退ステップにある時に，次に新たな起原集合と行線積を生成するために，どの行線を採用するかを決定するためのものである。このとき採用する行線の番号を nx で表す

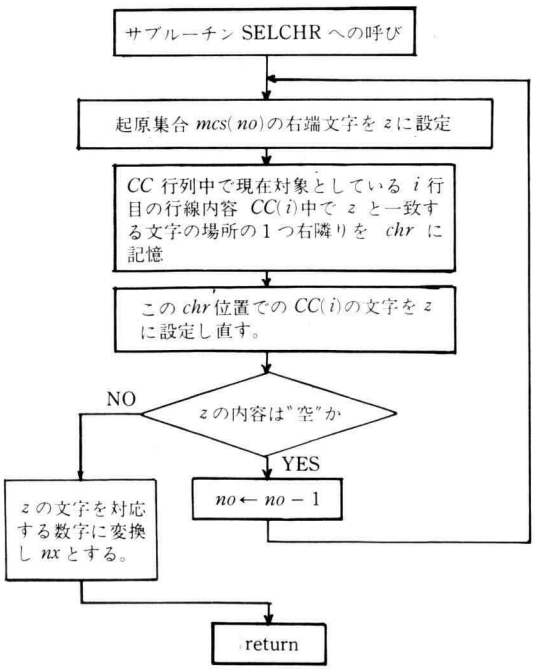


図3.　サブルーチンSELCHRのフローチャート

こととする。また，このサブルーチンのフローチャートを図3に示す。

図3より明らかなとおり，新たな起原集合と行線積を作るため，現在作られている起原集合の中の最右端文字よりも順番が後の文字，すなわち右側の文字が，現在対象としている i 回目の計算で扱う CC 行列の i 行目行線内容 $cc^{(i)}$ 中に存在しているか否か調べる。もし存在すれば，それが $cc^{(i)}$ の内容での順序番号ではなく，内部状態の順序番号 nx としてどこにあるかを決定して，この値を戻す。もし存在しなければ，行線積順序番号 no を1つ前に戻してこのアルゴリズムを繰り返す。

3.2　計算例

計算例を示すため，便宜上，図4の CC 行列を用いる。この計算例をいくつかの段階にまとめて示す。

［第1段階］　ステップ2で，内部状態数 n として4を，CP 数 $numcp$ として4を，さらに CP として ab, bc, bd および cd を入力する。この結果 CC 行列の各文字列は

$$\left.\begin{aligned} &cc^{(1)}=ab,\quad cc^{(2)}=abcd,\\ &cc^{(3)}=bcd,\ cc^{(4)}=bcd \end{aligned}\right\} \qquad (19)$$

となる。

［第2段階］　ステップ4で内部状態番号 i に対し1を設定し，ステップ5で行線積 $mcp^{(1)}$ と起原集合 $mcs^{(1)}$ は，

$$mcp^{(1)}=cc^{(1)}=ab,\ mcs^{(1)}=a \qquad (20)$$

となる。ステップ6で行線積更新回数 no を1に設定

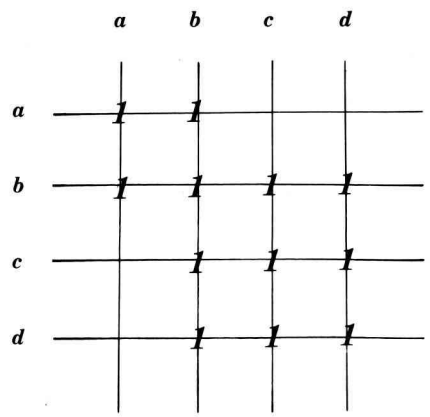


図4.　CP が ab, bc, bd および cd の場合の CC 行列

してサブルーチン SELCHR を呼ぶ。$mcs^{(1)}$ の右端文字 a の右隣りの文字が CC 行列の1行目文字列 $cc^{(1)}$ 内にあるか否かを調べ，そのアルファベット番号として，$nx=2$ を定める。この結果，ステップ7で CC 行列の nx 行目の文字列 $cc^{(nx)}=cc^{(2)}$ と内部状態アルファベット $its^{(nx)}=its^{(2)}$ が選択され，

$$\left.\begin{aligned} &mcp^{(2)}=mcp^{(1)}\wedge cc^{(2)}=(ab)\wedge(abcd)=ab,\\ &mcs^{(2)}=mcs^{(1)}\cdots its^{(2)}=a\cdots b=ab \end{aligned}\right\} \qquad (21)$$

が得られる。さらにステップ8で $mcp^{(2)}$ には有って $mcs^{(2)}$ には無い文字 rmd および $mcp^{(2)}$ には無くて $mcs^{(2)}$ には有る文字 $rmd1$ を調べると，共に"空"，すなわち

$$rmd=\text{``\ ''},\ rmd1=\text{``\ ''} \qquad (22)$$

となる。また，ステップ9で no を書き直して $no=no+1=1+1=2$ とする。

［第3段階］　ステップ10で式(22)が成立することから $mcp^{(2)}$ と $mcs^{(2)}$ の中味が全く等しいことが分り，定理2によって MCC が得られたことが分る。そこでステップ11で $mcp^{(2)}=ab$ が MCC として記憶される。ここで，$no=2$ としてサブルーチン SELCHR を呼び，$mcs^{(2)}$ の右端文字 b の右隣りの文字が $cc^{(1)}$ にあるか調べる。しかし，この文字が無いので no を1だけデクリメントして $mcs^{(1)}$ の右端文字 a のさらに右隣りの文字が $cc^{(1)}$ の中に存在するか調べる。この文字は存在し，その文字のアルファベット順序番号として $nx=2$ が求まる。しかしステップ16で no が1だけデクリメントされ，$no=0$ となり，ステップ17で $no<1$ となり，ステップ4に移って新たな i が設定されるので，この nx の値は結局使用されない。

［第4段階］　ステップ4で内部状態番号 i が新たに2に設定され，ステップ5で $mcp^{(1)}=abcd$, $mcs^{(1)}=b$ が設定される。ステップ6でサブルーチン SELCHR によりアルファベットに文字の順序番号 $nx=3$ が定まり，ステップ7で $mcp^{(2)}=mcp^{(1)}\wedge cc^{(3)}=bcd$, $mcs^{(2)}=mcs^{(1)}\cdots its^{(3)}=bc$ が得られる。このため，ステップ8で $rmd=d$ および $rmd1=$"　"が得られ，ステップ9で $no=2$ となる。

［第5段階］　ステップ10では，ステップ8で得られた条件からステップ12へ，さらにステップ12では，$rmd1=$"　" の条件よりステップ13へ進む（$rmd1=$"　"の条件は定理3の条件を満足しないからステップ15へ行けないのは当然である）。ステップ13では

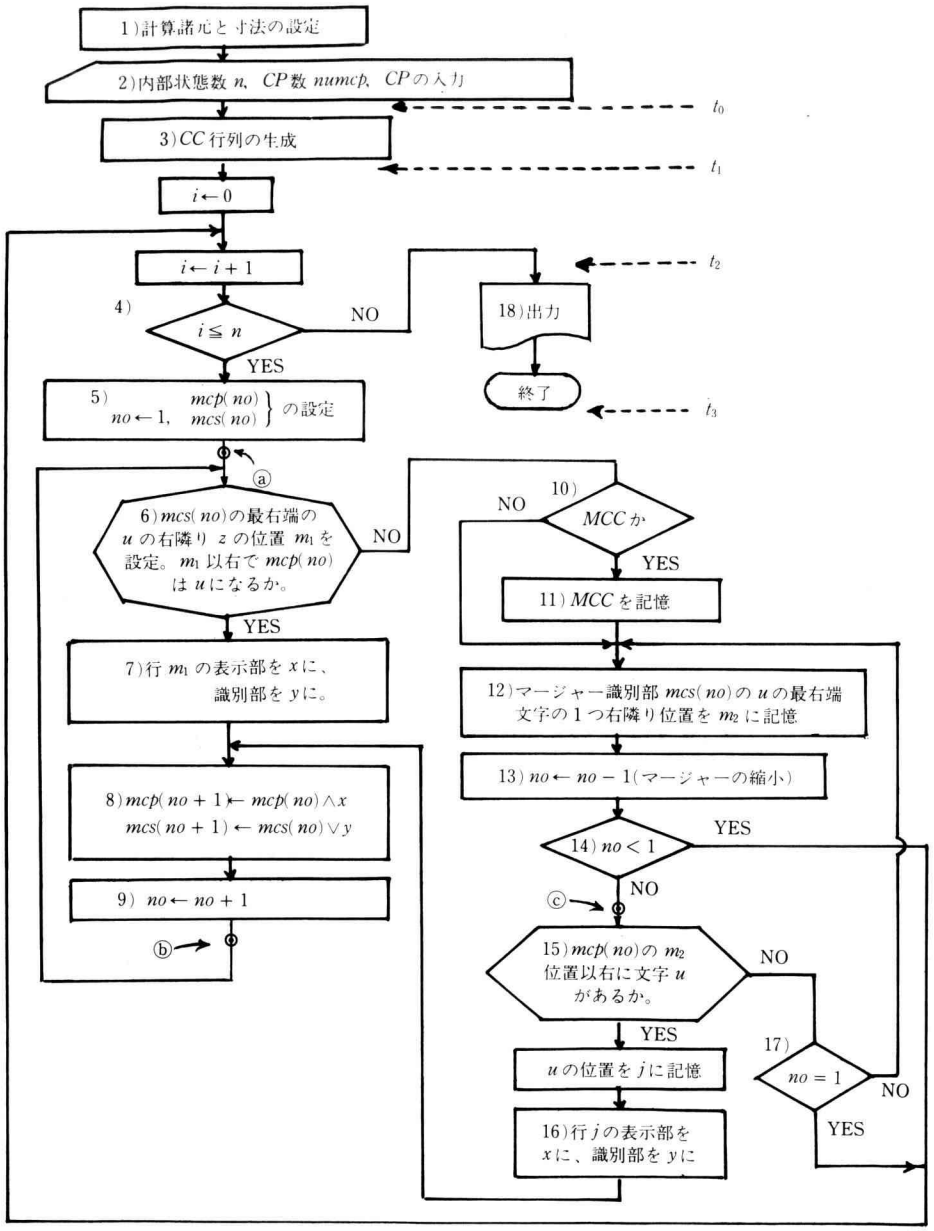


図5. Stoffersのアルゴリズムのフローチャート

rmd の右端文字 *d* は $mcs^{(2)}=bc$ の右端文字より右側にあり，定理4によってさらに高次の行線積が得られる可能性があるから，ステップ14へ進む。ステップ14ではサブルーチンSELCHRにより次に $mcs^{(2)}$ に連接すべきアルファベット文字の順序番号 *nx* として4を得る。ステップ7で $no=21\geq 1$ であるのでステップ7へ進む。

［第6段階］ ステップ7では，第2段階で述べたものと同様の手順により，$mcp^{(3)}=mcs^{(3)}=bcd$ が得られ，ステップ8，9，10および11を経て $mcp^{(3)}=bcd$

表1．7つの問題にたいするCPおよびMCC

問題番号	内部状態数	CP		MCC
		名　　称	個　数	
1	8	*ab, ad, ae, ag, bc, bd, be, cd, cf, cg, de, dh, eh, fg*	14	*deh, cfg, bcd, ag, abde*
2	6	*ab, ad, ae, bd, be bf, cd, cg, ef*	9	*cg, cd, bef, abe, abd*
3	6	*ab, ac, af, bc, bd, bf, cd, cf, de*	9	*de, bcd, abcf*
4	6	*ab, ad, af, bc, be, bf, cd, cf, de, ef*	10	*de, cd, bef, bcf, ad, abf*
5	8	*ab, ac, ad, ae, af, bd, bg, bh, cd, ce, cf, ch, dg, dh, ef, eg, fg, gh*	18	*efg, cdh, bdgh, acef, acd, abd*
6	6	*ab, ac, ad, bc, bd, be, cd, cf, ef*	9	*ef, cf, be, abcd*
7	9	*ab, ac, ae, af, ah, bc, be, bf, bg, bh, bi, cf, cg, ch, ci, de, df, dg, dh, di, ef, eg, eh, ei, fi, gh, gi, hi*	28	*deghi, defi, beghi, befi, bcghi, bcfi, abeh, abef, abch, abcf*

がMCCとして記憶される。さらにステップ15，16を経てアルファベット順序番号 nx が4に，また行線積更新番号 no はデクリメントされて1に設定される。この段階で $mcs^{(1)}=b$ となる。またステップ7で $no\geq 1$ となるので，再度ステップ7に戻る。

［第7段階］　以上のような手順を繰り返すことにより，以後 $mcs^{(no)}$ は，$mcs^{(2)}=bd$，$mcs^{(1)}=c$，および $mcs^{(2)}=cd$ の順に変化し，終了する。

4. Stoffersのアルゴリズム

Stoffersのアルゴリズムについては文献［6］および［13］で述べてあるが，その理論および詳細なアルゴリズムは不明である。ここでは，これらの文献の内容と，特に文献［6］に示されている計算過程の時系列的変化の図からStoffersのアルゴリズムを推定した。その手順を述べる前に術語の定義を行う。

［定義9］　CC行列の各行線相互間の同じ列要素同士でAND演算を行って得られる0と1とより成る新たな集合をマージャー表示部と呼び，この演算で採用した行線を表すのに，その行線に対応する列線上のみに1を置き，他は0とすることによって得られる0と1の要素の集合をマージャー識別部と呼ぶ。さらにこれらのマージャー表示部とマージャー識別部を統合してマージャーと呼ぶ。ここで，マージャーの生成順序回数を no で表し，マージャー表示部を $mcp^{(no)}$ で，マージャー識別部を $mcs^{(no)}$ で表すこととする。

つぎに，Stoffersのアルゴリズムを各ステップごとに述べる。また，そのフローチャートを図5に示す。

［ステップ1］　計算のための諸元および大きさの設定を行う。

［ステップ2］　内部状態数 n，CPの個数 $numcp$ およびすべてのCPを入力する。

［ステップ3］　CC行列の各行線と列線の交点が，0

表2．Stoffersアルゴリズムと本論文のアルゴリズムによる表1の7問題にたいするMCC生成演算時間（秒）の比較

（単位：秒）

問題番号		1	2	3	4	5	6	7
内部状態数		8	6	6	6	8	6	9
CP数		14	9	9	10	15	9	28
演算時間	Stoffers法	4	2	2	2	5	2	11
	本論文の方法	2	1	2	1	2	2	7

```
** CC **
a  ab
b  abcd
c  bcd
d  bcd

** MCC**
   ab
   bcd

** MERG/MCS **
  1  a
  2  ab
  3  a
  4  b
  5  bc
  6  bcd
  7  bc
  8  b
  9  bd
 10  b
 11  c
 12  cd
 13  c

** MERG/MCP **
  1  ab
  2  ab
  3  ab
  4  abcd
  5  bcd
  6  bcd
  7  bcd
  8  abcd
  9  bcd
 10  abcd
 11  bcd
 12  bcd
 13  bcd

****TIME*****
start time =05:49:21
mid-1 time =05:49:21
mid-2 time =05:49:23
  end time =05:49:23
```

（a） Stoffers 法

```
** CC **
a  ab
b  abcd
c  bcd
d  bcd

** MCC**
   ab
   bcd

** MERG/MCS **
  1  a
  2  ab
  3  b
  4  bc
  5  bcd
  6  b
  7  bd
  8  c
  9  cd

** MERG/MCP **
  1  ab
  2  ab
  3  abcd
  4  bcd
  5  bcd
  6  abcd
  7  bcd
  8  bcd
  9  bcd

****TIME*****
 start time  =14:21:51
 mid-1 time  =14:21:51
 mid-2 time  =14:21:52
   end time  =14:21:52
```

（b） 本論文の手法

図6. 図4のCPにたいする2つのプログラムのRUN結果

であれば“z”，1であれば“u”を与えるようにし，各行線ごとに行線内容 $cc^{(i)}$ を“z”と“u”の文字列として生成する。

［ステップ4］ MCCの最初の内部状態文字が，最初の文字から数えて i 番目にあるとき，i が内部状態数 n より小であればステップ5に移り，そうでなければステップ18に移る。

［ステップ5］ マージャー生成順序回数 no を1に設定し，サブルーチンROWSTを用いてCC行列の行 i の行線内容 $cc^{(i)}$ をマージャー表示部 $mcp^{(no)}$ に，またその行線位置をマージャー識別部 $mcs^{(no)}$ に設定する。

［ステップ6］ 識別部 $mcs^{(no)}$ で，その右隣りが“z”となるような“u”の中で最右端のものを探し（かりに左から n 番目，すなわち内部状態の終の位置が“u”になっている場合でも，$n+1$ 番目に“z”を強制追加するようにしてあるので，このような場合はすべて存在し得る），これを左から m_1 番目の位置とする。このような m_1 位置以右をマージャー表示部 $mcp^{(no)}$ について調べ，“u”が現れるか，“z”が現れるか確認する。

“u”が現れればステップ7に移り，最後まで“u”が現れず，“z”のままであればステップ10に移る。

［ステップ7］　サブルーチンROWSTを用いて，CC行列の行m_1の行線内容$cc^{(m_1)}$をマージャー表示部xに，またその行線位置に相当する列位置のみに“u”を持つ文字列をマージャー識別部yに設定する。

［ステップ8］　マージャー表示部$mcp^{(no)}$とxとの相対応する列要素間でAND演算を，すなわち相対応する両要素が共に“u”のときのみ“u”を与え，そうでないときは“z”を与える演算を行わせる。また，マージャー識別部$mcs^{(no)}$とyの相対応する要素間でOR演算を，すなわち相対応する両要素が共に“z”のときのみ“z”を与え，そうでないときは“u”を与える演算を行わせる。ここで次のマージャー生成順序回数は$no-1$となるので，表示部演算結果と識別部演算結果をそれぞれ$mcp^{(no+1)}$と$mcs^{(no+1)}$に格納する。

［ステップ9］　マージャー生成順序回数$no+1$をnoに書き直す（これはnoのインスクリメントではない）。

［ステップ10］　マージャー表示部$mcp^{(no)}$とマー

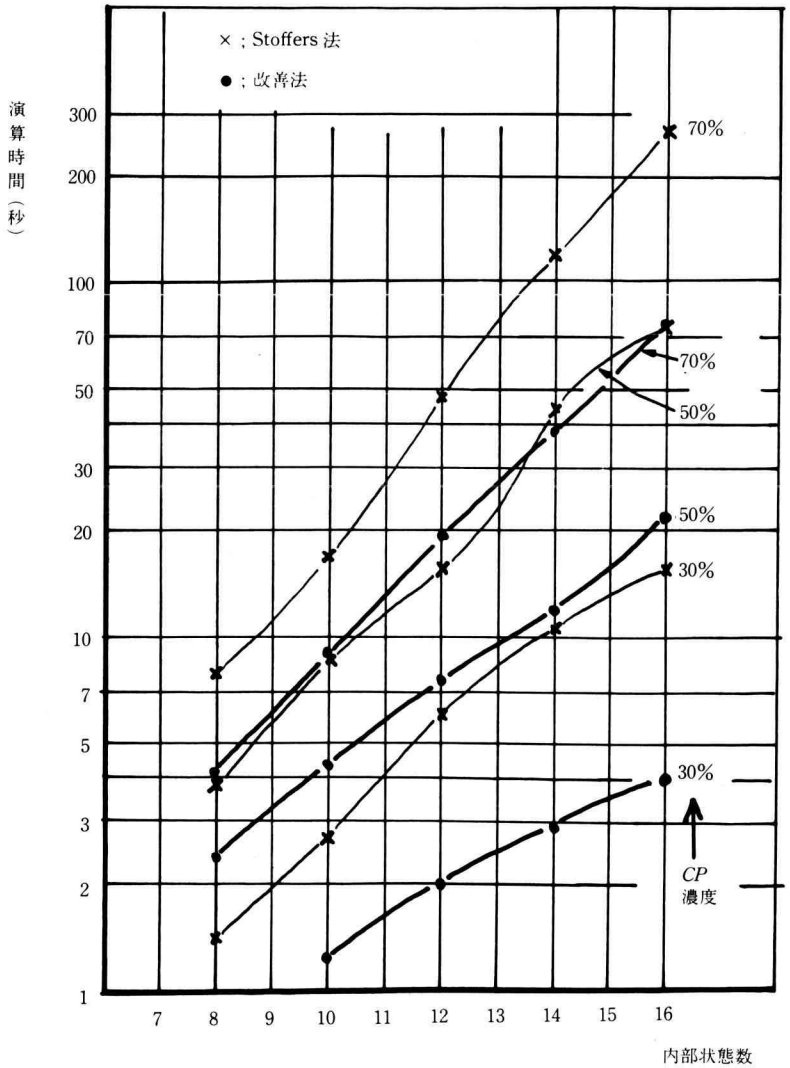


図7.　3つのCP濃度による内部状態数とMCC生成演算時間の関係

ジャー識別部 $mcs^{(no)}$ の文字列を左端から比較し，すべて同じかどうか確かめる。もしすべて同じであればステップ11に移る。1個でも同じでないものがあればステップ12に移る。

［ステップ11］　求まったマージャー表示部 $mcp^{(no)}$ をそのままMCCとして記憶する。

［ステップ12］　マージャー識別部 $mcs^{(no)}$ の文字が“u”となるものの中で最右端にあるものを検出し，それよりも1つの右寄りの位置を m_2 として記憶する。

［ステップ13］　マージャー生成順序回数 no を1だけデクリメントし，マージャーの縮小を行う。

［ステップ14］　もし，$no<1$ ならば，ステップ4へ戻る。そうでなければ次を行う。

［ステップ15］　縮小されたマージャーの表示部 $mcp^{(no)}$ 内の位置 m_2（すなわち，ステップ12で記憶した位置）以右に“u”があるか否か調べ，もしあればこの位置 j を記憶し，ステップ16に移る。無ければステップ17に移る。

［ステップ16］　サブルーチンROWSTを用いてCC行列の行 j の行線内容をマージャー表示部 x に，またその行線位置をマージャー識別部 y に設定し，ステップ8に移る。

［ステップ17］　マージャー生成順序回数 no が1より大ならば，ステップ12に戻る。もし no が1ならばステップ4に戻る。

［ステップ18］　ステップ17で i が n を越せば，作業を終り，諸データを出力する。

なお，サブルーチンROWSTでは，与えられた行番

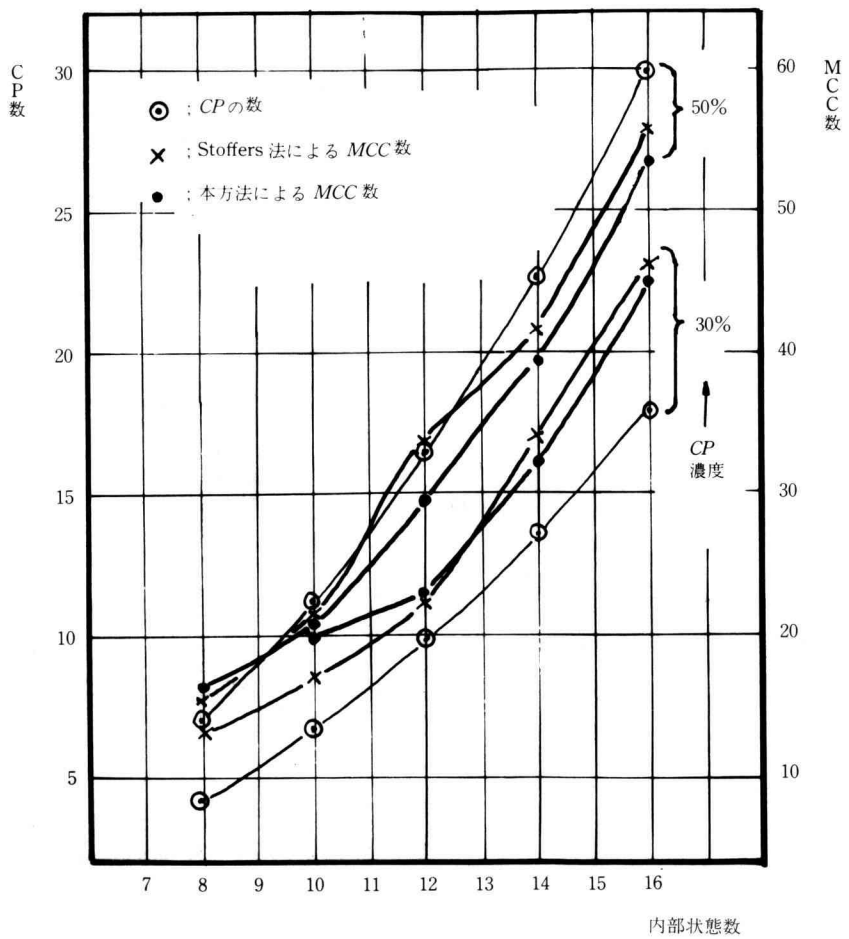


図8．　内部状態数，乱数CP入力数および生成MCC数の関係にたいするStoffers法と本方法の比較

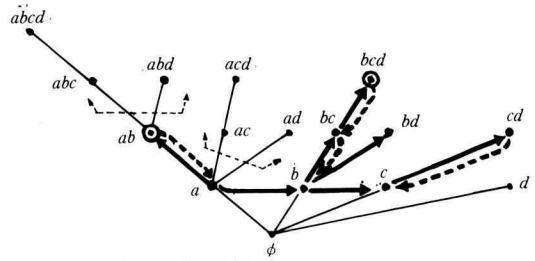
号にたいするCC行列中の文字列内容をxに設定し，この行番号と同じ列番号位置に“u”を持ち，他の位置ではすべて“z”を持つ文字列をyに設定する。

5. 演算結果と検討

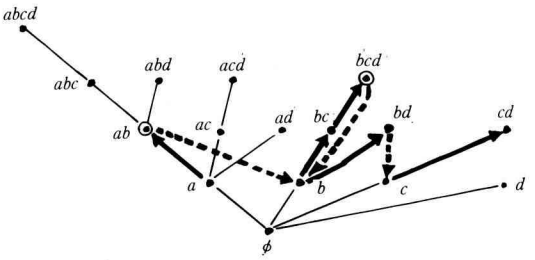
前章で述べたStoffersのアルゴリズムと本論文のアルゴリズムに従ってBASIC言語プログラムを作成し，パーソナルコンピュータPC9801VX（NEC製）上で実行させ，演算時間の比較を行った。なお，BASIC言語プログラムの作成に当っては，両アルゴリズム共，文字列操作命令を用いた。

表1は，不完全指定順序回路の最小化に当っていくつかの論文で引用されている7つの状態遷移表[9~12]から求めたCP，それらの個数および生成されるはずのMCCを示している。表1のようなCPを与える7つの場合を，それぞれ与えられた7つの問題と考え，各問題にたいして計算機を用いて2つのアルゴリズム（すなわちStoffers法と本論文の方法）によるプログラムを実行させてMCCを算出し，それに要した演算時間を測定した。その結果を表2に示す。なお，演算時刻は図2と図5のフローチャート上の位置t_0，t_1，t_2およびt_3で測定し，演算時間としてはt_3-t_0を採用した。この結果から，本論文の手法による演算時間はStoffersの手法によるものの1/2まで短縮できることが分った。

図6は，図4のCC行列で与えられる4つのCP，すなわちab，bc，bdおよびcdを入力として，上述した2つのプログラムを用いてMCCを生成した場合の出力結果を比較したものである。ここで，CCはCC行列を，MERG/MCSは本論文の起原集合，またはStoffers法のマージャー識別部を，さらにMERG/MCPは本論文の行線積，またはStoffers法のマージャー表示部を表す。これらは共に図2の本論文の手法のフローチャート，および図5のStoffers法のフローチャート上の ⓐ 点，ⓑ 点，および ⓒ 点で測定された。これらの3つの点は，両手法のアルゴリズムが同じ機能を与える点として選ばれた。すなわち，ⓐ 点は各行ごとの演算開始の初期設定を与える点であり，ⓑ 点は，本論文の場合は高次行線積生成演算後の点であり，Stoffers法では論理積・和演算後の点である。またⓒ点は，後退探索時にnoををデクリメントした後，$no\geq 1$を与える点として選ばれた。測定結果によれば，MERG/MCS（またはMERG/MCP）の測定回数



（a）Stoffers法による探索経路



（b）本手法による探索経路

図9．stoffers法と本手法による採素経路の比較

はStoffers法で13回，本論文の手法で9回となり，本論文の手法の方が探索回数が少なくなっている。

さらに2つの手法の性能の相異を明確にするため，内部状態数がさらに多い8～16個の範囲で，異なるCP濃度30%，50%および70%の場合についてMCC生成の演算時間を測定した。これを図7に示す。ここでCPの濃度ηは，CPの個数をn_{CP}，内部状態数をnとして，

$$\eta=2n_{CP}\times 100\div\{n(n-1)\}\ (\%) \qquad (23)$$

で表される。測定に当っては，n_{CP}個の乱数それぞれに対応するCPを用いてMCCを生成し，これを異なる乱数により10回繰り返し，各回の演算時間の平均をとった。これは，CPの構成に依存してMCC生成の難易度が変るのを平均化するためである。図7の結果より，本論文の手法による演算時間がStoffersの手法の場合の1/2以下まで短くできることが明らかである。特にCP濃度の低い方が，また内部状態数の多い方が，本論文の手法による演算時間の改善度が高い。内部状態の少ない方ではほぼ1/2まで，また多い方では1/3～1/4まで本論文の手法により演算時間が改善される。たとえばCP濃度が50%のとき，Stoffers法では10内部状態数で8.5秒，16内部状態数で76秒を要したものが，本論文の手法ではそれぞれ4.3秒と22秒に

改善された。

図8は，内部状態数を8～16とし，CP濃度30%と50%で図7の場合と同様にCPを入力し，生成されたMCCの平均数を上述の2つの手法について測定したものである。この図から，MCCの数については両手法ともほぼ同数が測定されたことが分る。

6. Stoffers法と本論文の方法との比較

与えられたいくつかのCPからMCCを生成する場合にいかにして演算時間を減少させるかが問題である。Stoffers法では，同じMCCを2回以上生成することを避け，内部状態を表すアルファベットの組合せを辞書編集順序（lexicographic）で探索するようにして演算時間を減らそうとしている。しかし，このためには次に示すような前進，後退の手順を明らかにしておく必要がある（Stoffersはこれを明らかにしていない）。

(1)　前進1回毎に次の順番の文字を右に連接する。

(2)　最右端の文字が順番の最後のものになると前進を止めて2回後退する。1回後退するごとに最右端文字を1個消去する。

(3)　このようにして2回後退し，現位置に到達したら1回前進する。このとき，現位置の1つ前の位置（すなわち(2)の状態から1回後退した位置）で持つ右端文字よりも1つ後の順番の文字を現位置の文字列の右端に連接する。

実際には，上述の手順で得られる探索回数以下に探索回数を削減されることが望ましい。その一つとして，上述の手順の中で，連接する最右端文字が順番の最後にならない中に，前進から後退に移ることが得策であり，その判定基準を設けることが望ましい。このため両手法ともCC行列を利用している。Stoffers法ではこのCC行列からマージャー表示部と識別部を作り，識別部の列要素と表示部の対応する列要素を比較して案内要素(leading element)を決定し，元のマージャーと案内要素で定まる新たな行との間で論理和と論理積演算を行うことにより新マージャーを生成するようにしている。この場合案内要素は前進探索のための指令要素である。しかしこの手法ではMCCが生成された後，およびMCCが生成される可能性のない時について，後退判定のアルゴリズムが明確でない。これにたいし本論文の手法ではStoffers法のマージャー表示部の代りに行線積を，マージャー識別部の代りに起原集合を用い，双方の集合の包含関係から証明される定理2，3および4を導入し，これらによって判定基準を明確にしている。

さらに本論文の手法は，Stoffers法に比し探索領域が減少する傾向を持つ。これは，本論文の手法が，特に後退探索に際し，探索ノード間を跳躍遷移する傾向を持つためである。図9はそれを説明する一例であり，図4のCC行列の示す4個のCP（*ab*, *bc*, *bd* および *cd*）が入力された結果得られる探索ノードとその経路をStoffers法と本論文の手法について比較している。実際の縮小探索経路を全探索経路上に太線で示してある。この図は図6の実験結果のMERG/MCSを基にしたものである。図6より通過するノード数はStoffers法で13個で，本論文の手法では9個に減少したことが分る。この探索範囲の縮小は次の理由による。すなわち，(1) 後退探索の手順が一元化されて，後退判定後のフローチャート上の後退経路は同じ経路（すなわちステップ15と16）を通るように整理されたこと，(2) 前進・後退の判定規則が定理2，3および4によって明確にされたこと（ステップ10，12および13で前進か後退かを明確に判定し，ステップ14を通じて前進が，ステップ15と16を通じて後退が行われること），および(3) サブルーチンSELCHRは前進・後退の両方にたいし次の行選択を行うようになっており，特に後退の場合，適切な行が見つかるまで跳躍的な行選択が行えるよう配慮されたこと，等である。

7. むすび

本論文では，不完全指定順序回路のMCCの生成に，縮小探索を行うCP依存法を適用する方法として提案されているStoffers法を改善する方法を提案した。この方法は組合せるべき内部状態の前進・後退に明確な判定基準を導入している。本論文の手法とStoffers法を，7つの代表例について，文字列操作命令を用いたBASIC言語プログラムでMCC生成の演算時間を比較した。また，3種類のCP濃度（30%, 50%, 70%）についても擬似的なCPを入力して同様な比較を行った。この結果，本論文の手法は，Stoffers法に比し，演算時間を1/2～1/4まで短縮した。このような演算時間の短縮は本論文の手法の3つの判定定理が有効であることを物語っており，これらの定理による探索経路の減少が有効であったものと推定される。特に本論文では跳躍的な探索が行われるのが大きな特徴である。

なお，筆者らはICPを用いたMCCの生成法についても種々検討を行っており，本論文のCP依存法とICP依存の各手法との比較についても今後検討を続けたい。また，他のプログラミング言語や計算機による種々の比較が残されている。

謝　辞

本研究のアルゴリズムを実現するプログラムを作成された平成元年度卒業研究生の三枝恭君と平成二年度卒業研究生の三上圭介君に謝意を表する。

参考文献

1) Paul, M.C. and Unger, S.H.: Minimizing the number of states in incompletely specified sequential switching functions, IRE Trans. Electron. Comput., Vol. EC-8, pp. 356-367 (1959).
2) Grasselli, A. and Luccio, F.: A method for minimizing the number of internal states in incompletely specified sequential networks, IEEE Trans. Electron. Comput., Vol. EC-14, pp. 350-359 (1965).
3) Choudhury, A.K., Basu, A.K. and DeSarkar, S.C.: On the determination of the maximum compatibility classes, IEEE Trans. Comput. (Corresp.), Vol. C-18, p. 665 (1969).
4) Sinha Roy, P.K. and Sheng, C.L.: A decomposition method of determining maximum compatibles, IEEE Trans. Comput., Vol. C-21, pp. 309-312 (1972).
5) Das, S.R.: On a new approach for finding all the modified cut-sets in an incompatibility graph, IEEE Trans. Comput., Vol. C-22, pp. 187-193 (1973).
6) Stoffers, K.E.: Sequential algorithm for the determination of maximum compatibles, IEEE Trans. Comput., Vol. C-23, pp. 95-98 (1974).
7) Biswas, N.N.: State minimization of incompletely specified sequential machines, IEEE Trans. Comput., Vol. C-23, pp. 80-84 (1974).
8) Rao, C.V.S. and Biswas, N.N.: Minimization of incompletely specified sequential machines, IEEE Trans. Comput., Vol. C-24, pp. 1089-1100 (1975).
9) 後藤：IMPCと両立性対の使用による不完全指定順序回路の最小化の一手法，情報処理学会論文誌，Vol. 28, No. 6, pp. 646-657 (1987).
10) 後藤，堺，巽：2進木利用による最大両立性クラスの作成，昭62，信学全大，564 (1987).
11) 後藤：主閉包集合の上限値設定による不完全指定順序回路の複数最小解の生成法，情報処理学会論文誌，Vol. 29, No. 9, pp. 873-887 (1988).
12) 後藤，巽，鈴木：不完全指定順序回路の最大両立性クラスの2分探索法による生成の検討，平成元，信学全大，A247 (1989).
13) Stoffers, K.E.: Scheduling of traffic lights — A new approach, Transportation Research, Vol. 2, pp. 199-234 (1968).