数理解析研究所講究録
161
第754巻 1991年 164-175

# RS型ベクトル機械の実際的応用の可能性について

九州大学工学部　岩本 宙造　(Chuzo Iwamoto)
九州大学工学部　岩間 一雄　(Kazuo Iwama)

## 1. まえがき

PRAMに代表される並列計算機は数多くのプロセッサと通信機能が複雑に組み合わされたシステムであって、その能力の仕組みを直観的かつ理論的に調べるためには必ずしも適切であるとは言えない。それに対しベクトル機械と呼ばれるモデルから古くから提案されており、並列計算機の能力が、より分かりやすいベクトル命令という形に凝縮されている。

ベクトル機械の命令セットとして、シフト演算[5]、連接、乗算[2]、乗算と除算[1]等が提案されており、それらはTMの多項式領域量を多項式時間量に加速する能力があることが知られている。また、シフト演算と除算は、$2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$（$n$重）時間を多項式時間に加速する能力があることが知られている。

文献[3]は、並列モデルの局所計算と通信をより明確に分離するため、ベクトル機械の命令セットとして、要素ごとの加減演算（局所計算に対応）や、ベクトルを拡大する2種類の *repeat*, *stretch* 演算ならびにそれらの逆演算（通信に対応）を導入した。*repeat* は、例えばベクトル $(a_1, a_2, \cdots, a_m)$ を $(a_1, a_2, \cdots, a_m, a_1, a_2, \cdots, a_m)$ に、*stretch* は $(a_1, a_1, a_2, a_2, \cdots, a_m, a_m)$ にする演算である。このベクトル機械（RS型ベクトル機械）は、上記演算の拡大縮小の度合（拡大率）を変化することにより、異なった能力のベクトル機械を作ることが可能であり、[3]はこのモデルがベクトル機械のある意味での標準形となりうることを主張した。より詳しくは、拡大率2のとき、TMの多項式領域を多項式時間量に、拡大率 $m$ のとき、交代性TMの指数時間多項式交代数を多項式時間量に加速する能力がある。さらに、文献[4]は、拡大率2は $k$ に、$m$ は $cm^k$ に置き換わることを示し、拡大率が指数的に大きくなったときの能力についても考察している。このようにRS型ベクトル機械の能力については、多項式時間という大雑把な範囲内での一般的シミュレーションに限定すればかなり明らかになったといえる。

本稿では拡大率が2のRS型ベクトル機械を用いて具体的なNP完全問題を高速に解くアルゴリズムを紹介する。NP完全問題には、例えば非決定性といった自明な並列アルゴリズムが存在するが、通信機能が強く制限されたRS型ベクトル機械上で議論することにより、興味深い（自明でない）アルゴリズムを提供することができる。本稿では、NP完全問題として、充足可能性問題とハミルトン閉路問題を扱う。RS型ベクトル機械は、各々を $O(n)$、$O(n\log^2 n)$ 時間で解くことができる。

**2.**でモデルの定義を行ない、**3.**でアルゴリズムの概要を説明し、**4.**で証明を簡潔にするための基本的サブルーチンを紹介する。**5.**で各証明の前半を、**6.**で後半を与える。

## 2. RS 型ベクトル機械

長さ $m$ のベクトルは $A=(a_1,a_2,\cdots,a_m)$ で表される。ここで各 $a_i$ ($A[i]$ でも表される) は、非負の整数 (スカラー) である。$A=(a_1,a_2,\cdots,a_i)$, $B=(b_1,b_2,\cdots,b_j)$、さらに $\circ$ をスカラーに対する 2 項演算とする。このとき、$A\circ B=(a_1\circ b_1,a_2\circ b_2,\cdots,a_k\circ b_k)$ (ただし、$k=\min(i,j)$) である (要素ごとの演算)。$A$ と $B$ の連接を $AB$ 又は $(A,B)$ と記し、$(a_1,a_2,\cdots,a_i,b_1,b_2,\cdots,b_j)$ を表す。$A^p=A^{p-1}A$, $A^1=A$ である。ベクトル $A$ の長さを $|A|$ と記す。$|A|=1$ のとき $(a_1)^l$ は、しばしば $a_1^l$ と書かれる。本稿では、ベクトル演算として *repeat, stretch* の 2 種類とそれぞれの逆演算 *fold, contract* を導入し、それぞれ $\downarrow$, $\rightarrow$, $\uparrow$, $\leftarrow$ で表す。$m$ をベクトルの長さとすると、これらの演算は拡大率を表す関数 $d(m)$ を与えることによって意味を持ち、$A=(a_1,a_2,\cdots,a_m)$, $B=(a_1,a_2,\cdots\cdots,a_{md(m)})$ に対し、次のように定義される。

$$\downarrow A=A^{d(m)}. \tag{1}$$

$$\rightarrow A=(a_1^{d(m)},a_2^{d(m)},\cdots,a_m^{d(m)}). \tag{2}$$

$$\begin{aligned}\uparrow B=(\ & f(a_1,\ a_{m+1},\ \cdots,\ a_{(d(m)-1)m+1}),\\ & f(a_2,\ a_{m+2},\ \cdots,\ a_{(d(m)-1)m+2}),\\ & \quad\vdots\quad\vdots\qquad\quad\vdots\\ & f(a_m,a_{2m},\ \cdots,\ a_{d(m)m}\qquad)).\end{aligned} \tag{3}$$

$$\begin{aligned}\leftarrow B=\ (\ & f(\ a_1,\ \ a_2,\ \ \cdots,\ a_{d(m)}\ ),\\ & f(\ a_{d(m)+1},\ a_{d(m)+2},\ \cdots,\ a_{2d(m)}\ ),\\ & \quad\vdots\qquad\qquad\qquad\vdots\\ & f(\ a_{(m-1)d(m)+1},\ \cdots,\ a_{md(m)}\ )).\end{aligned} \tag{4}$$

ただし、

$$f(a_1,a_2,\cdots,a_m)=\begin{cases}0 & \text{if } a_1=\cdots=a_m=0\\ a & \text{if 少なくとも一つの要素が } a \text{ で残りは } 0\\ \text{undefined} & \text{if 2つ以上の正の値を有する。}\end{cases} \tag{5}$$

拡大率 $d(m)$ の RS 型ベクトル機械 (以下 RS-VM) は RS-VM($d(m)$) で表され、それは以下に与える要素を使用できるプログラムとして定義される。

1. 有限個のスカラー変数、$x,y,z,\cdots$.
2. 有限個のスカラー定数、$a,b,c,\cdots$.
3. スカラー命令: $x:=y+z$, $x:=y-z(=0 \text{ if } z>y)$,

   **if** $(x>0)$ **goto** label, **accept**, **reject**.

4. 有限個のベクトル変数、$X,Y,Z,\cdots$.
5. 有限個のベクトル定数、$A,B,C,\cdots$.
6. 要素ごとのベクトル演算、$X:=Y+Z$, $X:=Y-Z$.

7. 拡大縮小ベクトル演算、 $X :=\downarrow Y,\ X :=\rightarrow Y,\ X :=\uparrow Y,\ X :=\leftarrow Y.$

8. ベクトル・スカラー変換、 $x := X[1].$

拡大率2と$k$の間に受理能力の差はないことから[4]、本稿では簡単のため拡大率$d(m) = 2$のRS-VMについてのみ考察する。ベクトル機械上で、より自然な議論を行なうため、本稿では、RS-VM(2)の扱うベクトルの長さは2のべきとする。これによって、扱う全てのベクトルに対し拡大縮小演算を適用することができる(長さ3のベクトルは縮小演算できない)。そこで、問題の入力も2のべきの長さのベクトルで与えるものとする。

## 3. アルゴリズムの概要

RS-VMでNP完全の問題を解く大雑把な流れは、次の(i), (ii)のようになる。(i)問題の解となりうる候補の全てを含むベクトルを作る。このベクトルを候補ベクトルと呼ぶ。候補ベクトルは、一つの解の候補を長さ$n$の部分ベクトルとして保持した構造を持つ。(ii) (i)で作った候補ベクトルに含まれる候補で問題の条件を満足する解を表すものが存在するかを調べる。本稿では以下で紹介する0/1割当型と順列型の候補ベクトルを使用する。

0/1割当型： これは、$x_1, x_2, \cdots, x_n$へそれぞれ0/1を割り当てるタイプである。この候補ベクトルを$C_{01}$と呼び、次に示すものである。

$$C_{01} = (\underbrace{(\overbrace{0\cdots00}^{n})^m(\overbrace{0\cdots01}^{n})^m\cdots(\overbrace{1\cdots11}^{n})^m}_{n\cdot m\cdot 2^n}) \tag{6}$$

長さ$n$の部分ベクトル（ブロックと呼ぶ）は$(x_1, \cdots, x_n)$への0/1の割当を表している。同じブロックの$m$回の繰り返し構造の理由は、(ii)の実行をより容易にするためである。$C_{01}$を用いる問題に充足可能性問題がある。

順列型： これは$1, 2, \cdots, n$を並び変えた全順列（$n!$ある）を長さ$n$の部分ベクトル（ブロック）として含んだベクトルである。このベクトルはハミルトン閉路問題に用いる。

以上の候補ベクトルは、本稿で論じる問題以外にも数多くのNP完全問題に適用可能であると考えられる。候補ベクトルの作成時間は、0/1割当型は$O(n)$時間、順列型は$O(n\log n)$時間である。拡大率が2のとき、長さ$l$のベクトルの作成に$\Omega(\log l)$時間かかることから、候補ベクトルの作成時間を改善するのはかなり困難と思われる。

多項式時間RS-VMは多項式領域TMと同じ能力を持つので、多項式時間でNP完全問題を解けることは自明である。しかし、$O(n\log^k n)$時間で解くアルゴリズムを示すのは容易ではなく、個々の問題に対して興味深いアルゴリズムを与えることができる。特に、上記の候補ベクトルの構成時間（これは改良の余地はないと思われる）より短い時間で(ii)の実行が可能かどうかは、興味ある問題の一つである。充足可能性問題に関しては候補ベクトルの構成時間以下で実行している。しかし、ハミルトン閉路問題は$O(\log n)$倍の時間に押さえるに留まった。これは、本稿で残された興味ある未解決問題の一つである。

## 4. 基本的サブルーチン

本節では、アルゴリズムを簡潔に記述するための基本的サブルーチンを紹介する。

要素毎の倍加演算： $A := A + A$を$i$回繰り返すことにより、$A$の要素$a$を$a \cdot 2^i$にすることができる。

反転演算： $A$ を 0 と $a$ なる要素から構成されるベクトルとする。

$$(\underbrace{a, a, \cdots, a}_{|A|}) - A \tag{7}$$

を実行することにより、$A$ の要素の 0 と $a$ を入れ換えることができる。こうして得られるベクトルを $\neg A$ で表す。

ブール化演算： $A$ を任意のベクトルとする。

$$(\underbrace{1, 1, \cdots, 1}_{|A|}) - A \tag{8}$$

を実行し、更に反転演算を適用すると、$A$ に含まれていた 0 でない要素を 1 に換えることができる。

要素毎の論理和演算 $\vee$： $A$ と $B$ を互いに同じ長さの 0/1 ベクトルとする。$A + B$ をブール化することにより $A \vee B$ が計算できる。

要素毎の論理積演算 $\wedge$： $A \wedge B = \neg(\neg A \vee \neg B)$.

要素毎の排他論理和演算 $\oplus$： $A \oplus B = (A \vee B) \wedge \neg(A \wedge B)$.

比較一致演算：ベクトル $A$, $B$ が同じ値を持つ場所を 1 として取り出そうとしているとする。まず、

$$D := (A - B) + (B - A) \tag{9}$$

で得られる $D$ をブール化し、更に反転演算を適用することにより得られる。

マスク演算：ベクトル $A$, $MASK$ を考える。但し、$MASK$ は 0/1 ベクトルとする。今、$MASK$ で $A$ をマスク、つまり、$MASK$ と $A$ の要素ごとの積を行なおうとしているとする。まず、$\neg MASK$ に対し倍加演算を適用し、$\neg MASK$ の 0 でない要素の値を十分大きくする。こうして得られるベクトルを $A$ から引くと、目的のベクトルが得られる。$A$ の要素の最大値を $m$ とすると、マスク演算は $O(\log m)$ で実行できる。

マスク演算から、新たな演算を導くことができる。例えばベクトル $C$ の要素で $c$ よりも値の大きい要素を全て 0 にしたいとする。まず最初に、$C$ の各要素から $c$ を引き、その結果に対しブール化演算、反転演算を行なう。こうして $MASK$ が作られる。目的のベクトルは $C$ を $MASK$ でマスクすることにより得られる。

$\wedge$-*contract* 演算：この演算は、(4) 式と次の

$$f(a_1, a_2, \cdots, a_m) = \begin{cases} 0 & \text{if 少なくとも一つの要素が 0 で残りは } a \\ a & \text{if } a_1 = a_2 = \cdots = a_m = a \\ \text{undefined} & \text{if 2 つ以上の正の値を有する。} \end{cases} \tag{10}$$

で定義される。このベクトル演算は、反転演算、*contract* 演算、反転演算と実行することにより実現できる。$\wedge$-*fold* 演算も同様に定義できる。

## 5. 候補ベクトルの作成

候補ベクトルの作成方法について述べる前に、二つの命題を証明する。

[命題 1 ]
$$NUM(j) = (0, 1, \cdots, j-1) \tag{11}$$

とする。$NUM(j)$ は $O(\log j)$ 時間で構成できる。
（証明）帰納的に作成する。基底段階としてベクトル定数 $(0,1)$ から始める。今、

$$NUM(2^{2^i}) = (0, 1, \cdots, 2^{2^i} - 1) \tag{12}$$

まで完成しているとする。$NUM(2^{2^i})$ を $2^i$ 回 *repeat* 演算し

$$(\overbrace{\underbrace{01\cdots(2^{2^i}-1)}_{2^{2^i}}\ \underbrace{01\cdots(2^{2^i}-1)}_{2^{2^i}}\cdots\ \underbrace{01\cdots(2^{2^i}-1)}_{2^{2^i}}}^{2^{2^{i+1}}}) \tag{13}$$

を得る。次に $NUM(2^{2^i})$ を $2^i$ 回 *stretch* し

$$(\underbrace{0\cdots0}_{2^{2^i}}\ \underbrace{1\cdots1}_{2^{2^i}}\cdots\ \underbrace{(2^{2^i}-1)\cdots(2^{2^i}-1)}_{2^{2^i}}) \tag{14}$$

を得る。(14) に倍加演算を $2^i$ 回適用し

$$(\underbrace{0\cdots0}_{2^{2^i}}\ \underbrace{(2^{2^i})\cdots(2^{2^i})}_{2^{2^i}}\cdots\ \underbrace{(2^{2^{i+1}}-2^{2^i})\cdots(2^{2^{i+1}}-2^{2^i})}_{2^{2^i}}) \tag{15}$$

を作る。$NUM(2^{2^{i+1}})$ は (13)+(15) で得られる。このように、$NUM(2^{2^i})$ から $NUM(2^{2^{i+1}})$ の構成は $O(2^i)$ ステップで得られる。

$NUM(j)$ は $NUM(2^{2^k})$ $(k = \lceil \log\log j \rceil)$ を構成し、これに

$$(\underbrace{00\cdots0}_{j}) \tag{16}$$

を加えることにより得られる。((16) はベクトルを長さ $j$ に切る目的に使用する。) $NUM(j)$ の作成時間 $T$ は

$$\begin{aligned} T &= \sum_{i=0}^{\lceil \log\log j \rceil} O(2^i) \\ &= O(\log j) \end{aligned} \tag{17}$$

となる。∎

[命題 2] ベクトル $C_{1l}$ を

$$C_{1l} = ((\underbrace{1\cdot\cdot11}_{n})^m(\underbrace{1\cdot\cdot12}_{n})^m\cdots(\underbrace{1\cdot\cdot1l}_{n})^m\cdots\cdots(\underbrace{l\cdot\cdot ll}_{n})^m) \tag{18}$$

とする。$C_{1l}$ は $O(n\log l)$ 時間で構成できる。
（証明）まず、

$$NUM'(j) := NUM(j) + (\underbrace{1\cdots1}_{j}) \tag{19}$$

とおく。$C_{1l}$ を、$h$ に関して帰納的に構成する。

基底段階：ベクトル定数 $NUM'(l)$ を $\log mn$ 回 *stretch* して

$$C_{1l}(1)=((\underbrace{1\cdot\cdot1\overbrace{1}^{1}}_{n})^m(\underbrace{2\cdot\cdot2\overbrace{2}^{1}}_{n})^m\cdots(\underbrace{l\cdot\cdot l\overbrace{l}^{1}}_{n})^m) \tag{20}$$

を得る。また、

$$(NUM(n))^{ml}, \tag{21}$$

$$(\underbrace{n-1,\cdots,n-1}_{n})^{ml}, \tag{22}$$

$$(\underbrace{1,1,\cdots,1}_{n})^{ml} \tag{23}$$

を作る。

帰納段階：今、

$$((\underbrace{1\cdots1}_{n\cdot l^{h-1}})^m(\underbrace{2\cdots2}_{n\cdot l^{h-1}})^m\cdots(\underbrace{l\cdots l}_{n\cdot l^{h-1}})^m), \tag{24}$$

$$(NUM(n))^{ml^h}, \tag{25}$$

$$(\underbrace{n-h,\cdots,n-h}_{n})^{ml^h}, \tag{26}$$

$$(\underbrace{1,1,\cdots,1}_{n})^{ml^h} \tag{27}$$

及び、次に示す $C_{1l}(h)$ が得られているとする。

$$C_{1l}(h)=\overbrace{((\underbrace{\cdot\cdot\overbrace{1\cdot\cdot11}^{h}}_{n})^m(\underbrace{\cdot\cdot\overbrace{1\cdot\cdot12}^{h}}_{n})^m\cdots(\underbrace{\cdot\cdot\overbrace{1\cdot\cdot1l}^{h}}_{n})^m\cdots\cdots(\underbrace{\cdot\cdot\overbrace{l\cdot\cdot ll}^{h}}_{n})^m)}^{nml^h} \tag{28}$$

$C_{1l}(h)$ は $h$ ビットの全ベクトル（$l^h$ ある）をブロックの下位 $h$ ビットに含んだベクトルである。(24) を $\log l$ 回 *stretch*、(28) を $\log l$ 回 *repeat* し、得られた二つのベクトルの各ブロックの右から $h+1$ 番目の要素を置き換えることにより $C_{1l}(h+1)$ を得る。$h+1$ 番目の取り出しは、

$$(\underbrace{0\cdots0\overbrace{10\cdots0}^{h+1}}_{n})^{ml^h} \tag{29}$$

を $\log l$ 回 *repeat* して得られるベクトルとマスク演算を用いることにより行なう（$O(\log l)$ 時間で可能）。(29) は、(26) から (27) を引いて、

$$(\underbrace{n-h-1,\cdots,n-h-1}_{n})^{ml^h} \tag{30}$$

を作り、これと (25) の比較一致演算により得られる。次の段階のために、(25), (26), (27) を各々 $\log l$ 回 *repeat* しておく。

以下、$h=n$ まで繰り返し、$C_{1l}$ を $O(n\log l)$ 時間で得る。∎

### 5.1 0/1 割当型

候補ベクトル $C_{01}$ は、命題 2 の方法で $C_{12}$ を $O(n)$ 時間で作り、

$$C_{01} := C_{12}-(\underbrace{1,1,\cdots,1}_{|C_{12}|}) \tag{31}$$

で得られる。

### 5.2 順列型

順列型の候補ベクトルを $C_P$ と呼ぶ。まず、次に示す $C_{1n}$ を $O(n\log n)$ 時間で作る。

$$C_{1n}=((\overbrace{1\cdots11}^{n})^m(\overbrace{1\cdots12}^{n})^m\cdots(\overbrace{1\cdots1n}^{n})^m\cdots\cdots(\overbrace{n\cdots nn}^{n})^m) \tag{32}$$

これに含まれるブロックで $1,2,\cdots,n$ を並び変えてできるブロックのみを残し、そうでないブロックを取り除いて $C_P$ を得る。$C_{1n}$ を $\log n$ 回 *repeat*、$NUM'(n)$ を $\log n^{n+1}$ 回 *stretch* して、二つのベクトル

$$(\overbrace{C_{1n}}^{n^{n+1}},\overbrace{C_{1n}}^{n^{n+1}},\cdots,\overbrace{C_{1n}}^{n^{n+1}}) \tag{33}$$

$$(\underbrace{11\cdots1}_{\text{区間 1}},\underbrace{22\cdots2}_{\text{区間 2}},\cdots,\underbrace{nn\cdots n}_{\text{区間 }n}) \tag{34}$$

をそれぞれ構成する。((34) の 1 区間の長さは $C_{1n}$ の長さと一致することに注意。) (33) と (34) の比較一致演算を行ない $\tilde{P}$ を作る。$\tilde{P}$ を $\log n$ 回 *contract* し、更に $\log n$ 回 *stretch* すると、一つでも要素 1 を含むブロックの要素の全てを 1 に換えることができる。ここで、1 から $n$ の要素を全て含む $C_{1n}$ のブロックは、全区間で $n$ ビットの 1 が立つ。このベクトルを $\log n$ 回 $\wedge$-*fold* し、ベクトル $P$ を作る。$P$ で $C_{1n}$ をマスクすることにより $C_P$ が得られる。

## 6. アルゴリズム

本節では、5.で構成した候補ベクトルを用いて各 NP 完全問題を解くアルゴリズムを与える。

### 6.1 充足可能性問題

充足可能性問題は、リテラル $x_j$, $\bar{x}_j$ $(j=1,2,\cdots,n)$ の和より成る節 $c_i$ $(i=1,2,\cdots,m)$ の全てを真にするような 0 と 1 の割当があるかどうかを問う問題である。

RS-VM への入力は、$n$, $m$ (2 のべき) と 2 つのベクトル $\mathcal{C}=(C_1\ C_2\ \cdots\ C_m)$, $\mathcal{N}=(N_1\ N_2\ \cdots N_m)$ で与えられる。ただし、$\mathcal{C}$ を構成する各部分ベクトル $C_i$ は、節 $c_i$ を長さ $n$ のベクトルに変換したもので、$c_i$ に $x_j$ または $\bar{x}_j$ が含まれるとき $C_i[j]=1$ である。$c_i$ に $\bar{x}_j$ が含まれるとき $N_i[j]=1$ とする。その他の要素は全て 0 である。例えば、$n=8$, $c_i=(x_2+\bar{x}_3+\bar{x}_5+x_8)$ のとき

$$\begin{aligned} C_i &= (01101001),\\ N_i &= (00101000) \end{aligned}$$

である。$n$ が 2 のべきでない時は、如何なる節にも含まれない架空のリテラルが必要なだけ存在するとして、$m$ が 2 のべきでない時は、$(x_1+x_2+\cdots+x_n)$ なる架空の節が必要なだけ存在するとして入力ベクトルを与える。$x_1=\cdots=x_n=0$ で充足可能かどうかは、特別な場合として別に確かめる。

[定理 1] RS-VM(2) は充足可能性問題を $O(n)$ 時間で解く。

(証明) まず、$C_{01}$ を $O(n)$ 時間で構成する。次に入力ベクトル $\mathcal{C}$, $\mathcal{N}$ をそれぞれ $n$ 回 *repeat* し、それぞれ $\mathcal{C}_r$, $\mathcal{N}_r$ とおく。$C_{01}$, $\mathcal{C}_r$, $\mathcal{N}_r$ はそれぞれ次のような構造を持つ。

$$\begin{array}{lccccccccccc} C_{01}= & \overbrace{0\cdots00}^{n}, & \cdots, & \overbrace{0\cdots00}^{n}, & \overbrace{0\cdots01}^{n}, & \cdots, & \overbrace{0\cdots01}^{n}, & \cdots\cdots\cdots, & \overbrace{1\cdots11}^{n}, & \cdots, & \overbrace{1\cdots11}^{n}\\ \mathcal{C}_r= & C_1 & \cdots & C_m & C_1 & \cdots & C_m & \cdots\cdots\cdots & C_1 & \cdots & C_m\\ \mathcal{N}_r= & N_1 & \cdots & N_m & N_1 & \cdots & N_m & \cdots\cdots\cdots & N_1 & \cdots & N_m. \end{array} \tag{35}$$

ベクトル $\mathcal{N}_r$ に1が立っている要素に対応するベクトル $C_{01}$ の要素の0と1を反転させて、$\widehat{C}_{01}$ とおく。これは、

$$\widehat{C}_{01} := C_{01} \oplus \mathcal{N}_r \tag{36}$$

で計算される。0/1の割当が各節を真にするかを調べるため、$\widehat{C}_{01}$ と $C_r$ の要素ごとの論理積演算を行なう。この時得られるベクトルの各ブロック内に少なくとも一つ1が立っていれば、そのブロックが表す節は充足されることになる。これを調べるため $\log n$ 回 *contract* 演算を行なうと、ブロックに対応する $n$ ビットの論理和をとったことになる。次に、隣合う $m$ 個のブロックの全てが充足されているかを調べるために $\log m$ 回 $\wedge$-*contract* する。この時得られるベクトルの要素に少なくとも一つ1が立っていれば、ある割当が $c_1, c_2, \cdots, c_m$ の全てを真にすることを意味する。つまり、更に $n$ 回 *contract* 演算を行なって長さ1のベクトルを作り、このベクトルの要素が1であれば問題の解は yes である。∎

### 6.2 ハミルトン閉路問題

ハミルトン閉路問題は、与えられた無向グラフ $G$ に対し全ての頂点をちょうど一回通過し出発点に戻る道が存在するかを問う問題である。

RS-VM への入力は、$n \times n$ の0と1の要素を持つ接続行列 $I$ で与えられる。

$$I = \begin{pmatrix} e_{11}, & e_{12}, & \cdots & e_{1n}, \\ e_{21}, & e_{22}, & \cdots & e_{2n}, \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1}, & e_{n2}, & \cdots & e_{nn} \end{pmatrix} \tag{37}$$

ここで、$e_{ij}$ は、その値が1のとき頂点 $i$ と $j$ 間に枝が存在することを意味する。このベクトルは分かり易いように正方形の行列状に表したが、実際は単なる1次元ベクトルであることに注意されたい。また、$n$ は2のべきとするが、そうでない時は、$G$ の任意の頂点 $i$ を選ぶ。これに隣接する頂点を $j_1, j_2, \cdots, j_k$ とする。$G$ に架空の頂点 $i_1, i_2, \cdots, i_l$ が必要なだけ存在するとし（但し、$l \geq 2$ である）、$(i, i_1), (i_1, i_2), \cdots, (i_{l-1}, i_l)$、$(j_1, i_l), (j_2, i_l), \cdots, (j_k, i_l)$ なる架空の枝が存在するとしてベクトル化する。

[定理3] RS-VM(2) はハミルトン閉路問題を $O(n\log^2 n)$ 時間で解く。

（証明） まず、5.で述べた候補ベクトル $C_P$ を構成する。但し、$m = n^{\log n}$ とする。各ブロックの $i$ 番目の要素が $j$ ならば頂点 $i$ から頂点 $j$ へ枝を辿ることを表す。例えば $n = 4$ のとき、(3421) は頂点 $1-3, 2-4, 3-2, 4-1$ なる枝を辿ることを意味する。言い替えると $1 \to 3 \to 2 \to 4 \to 1$ なる順序で頂点を辿ることを意味する。同じ部分ベクトルの $m$ 回の繰り返し構造の理由については後で述べる。

次に、$C_P$ に含まれるブロックで表わされた頂点を辿る順番のうち、入力グラフに対し妥当でないものを除く。まず $C_P$ の各ブロックを $n^2$ の行列に換える。$C_P$ を $\log n$ 回 *stretch* し、$(NUM'(n))^{m \cdot n^{n+1}}$ と比較一致演算してベクトル $MT$ を作る。例えば、あるブロックが (3421) であった時、$\log n$ 回 *stretch* されて

$$\cdots\cdots(3333\ 4444\ 2222\ 1111)\cdots\cdots \tag{38}$$

となる。これと

$$\cdots\cdots(1234\ 1234\ 1234\ 1234)\cdots\cdots \tag{39}$$

の比較一致演算を行ない

$$\cdots\cdots(\underbrace{0010}\ \underbrace{0001}\ \underbrace{0100}\ \underbrace{1000})\cdots\cdots \tag{40}$$

なるベクトルを作る。このベクトルを、

$$\begin{array}{l}(\ 0,\ 0,\ 1,\ 0,\\ \ \ 0,\ 0,\ 0,\ 1,\\ \ \ 0,\ 1,\ 0,\ 0,\\ \ \ 1,\ 0,\ 0,\ 0\ )\end{array} \tag{41}$$

なる閉路の接続行列とみなす。次に入力ベクトル $I$ を $\log mn^n$ 回 *repeat* し、$I^{mn^n}$ を作る。$MT[i]=1$ かつ $(I^{mn^n})[i]=0$ に限り $K[i]=1$ なるベクトル $K$ を作る。これは、

$$K := MT - I^{mn^n} \tag{42}$$

により容易に計算できる。$K$ を $\log n^2$ 回 *contract* して $\log n^2$ 回 *stretch* し $K'$ を得る。$\neg K'$ で $MT$ をマスクすることにより入力グラフに対し妥当でないものを除くことができる。

以上の操作で作られたベクトルを $MT^{(1)}$ と呼ぶ。$MT^{(1)}$ は $n^2$ 長で表された接続行列を $mn^n$ 回含んだ構造を持つ。この時点では、$MT^{(1)}$ はグラフ $G$ 上に長さ $n$ より小さい閉路を2個以上持つものも含まれていることに注意しなければならない。例えば、(1342) は、$1\rightarrow 1,\ 2\rightarrow 3\rightarrow 4\rightarrow 2$ の二つの閉路に分かれている。接続行列のうち長さが $n$ の閉路を一つ含んだ接続行列が存在すれば、それはグラフ $G$ のハミルトン閉路を表す。それを調べるために $MT^{(1)}$ の各接続行列を自乗してゆき、距離 $1, 2, 4, \cdots, n/2$ の接続行列を含んだベクトル $MT^{(1)}$, $MT^{(2)}$, $MT^{(4)}$, $\cdots$, $MT^{(n/2)}$ を順に作成していき、$n$ より短い2のべきの長さの閉路を含む接続行列（対角要素に少なくとも一つ1が立つ）を除いてゆく。最後に、長さ $n$ の閉路を選ぶため $MT^{(n)}$ を計算し、対角要素が全て1である接続行列を選ぶ。$n$ より短い2のべきの長さの閉路を含む接続行列は、$MT^{(n)}$ を計算するまでに必ず除いておかなければならない。（長さ $2^i < n$ の閉路は $MT^{(n)}$ において対角要素に1を立てることに注意。）

$MT^{(1)}$ から長さ1の閉路を含む行列の除去は、次のようにしてなされる。$n\times n$ の行列 $\downarrow NUM(n)$ と $\rightarrow NUM(n)$ を構成し、比較一致演算すると対角要素に1を立てた $n$ 行 $n$ 列の行列 $DIA(n)$ が得られる。さらに $\log mn^n$ 回 *repeat* して、$(DIA(n))^{mn^n}$ を得る。このベクトルで $MT^{(1)}$ をマスクしてベクトル $MK$ を作る。$MK$ を $\log n^2$ 回 *contract*、更に $\log n^2$ 回 *stretch* し、$MASK$ を作る。$\neg MASK$ で $MT^{(1)}$ をマスクすることにより実現できる。

次に、$MT^{(1)}$ の各行列を行列ごとに自乗し、距離2の接続行列を表す行列を部分ベクトルとして含んだ $MT^{(2)}$ を作る。この方法を説明する前に読者の理解を助けるため、長さ $n^2$ の接続行列

$$\begin{array}{rl}A = (\ a_{11}, & a_{12}, \ \cdots\ a_{1n},\\ \ \ a_{21}, & a_{22}, \ \cdots\ a_{2n},\\ \ \ \vdots & \ \ \vdots\ \ \ddots\ \ \vdots\\ \ \ a_{n1}, & a_{n2}, \ \cdots\ a_{nn})\end{array} \tag{43}$$

を自乗する方法を記述した命題を示し、その後 $MT^{(2)}$ の作り方を示す。

[ 命題3 ][3] RS-VM(2) は接続行列 $A$ の自乗を $O(\log n)$ 時間で行なう。

（証明） $A$ を1行 $n^2$ 列の行ベクトルと見なして、$\log n^2$ 回 *repeat* し、$n^2$ 行 $n^2$ 列のベクトル $A_r$ を作る。次に、$A$ を $n^2$ 行1列の列ベクトルと見なして、$\log n^2$ 回 *stretch* し、$n^2$ 行 $n^2$ 列のベクトル $A_s$ を作る。$A_r$ と $A_s$ の要素ごとの論理積をとり、次の (44) で $b^j_{ik}$ で示される

部分のみマスクして残す（(44) は $n=4$ の場合）。

$$\begin{array}{|cccc|cccc|cccc|cccc|}
\hline
b^0_{00} & & & & b^0_{10} & & & & b^0_{20} & & & & b^0_{30} & & & \\
b^0_{01} & & & & b^0_{11} & & & & b^0_{21} & & & & b^0_{31} & & & \\
b^0_{02} & & & & b^0_{12} & & & & b^0_{22} & & & & b^0_{32} & & & \\
b^0_{03} & & & & b^0_{13} & & & & b^0_{23} & & & & b^0_{33} & & & \\
\hline
 & b^1_{00} & & & & b^1_{10} & & & & b^1_{20} & & & & b^1_{30} & & \\
 & b^1_{01} & & & & b^1_{11} & & & & b^1_{21} & & & & b^1_{31} & & \\
 & b^1_{02} & & & & b^1_{12} & & & & b^1_{22} & & & & b^1_{32} & & \\
 & b^1_{03} & & & & b^1_{13} & & & & b^1_{23} & & & & b^1_{33} & & \\
\hline
 & & b^2_{00} & & & & b^2_{10} & & & & b^2_{20} & & & & b^2_{30} & \\
 & & b^2_{01} & & & & b^2_{11} & & & & b^2_{21} & & & & b^2_{31} & \\
 & & b^2_{02} & & & & b^2_{12} & & & & b^2_{22} & & & & b^2_{32} & \\
 & & b^2_{03} & & & & b^2_{13} & & & & b^2_{23} & & & & b^2_{33} & \\
\hline
 & & & b^3_{00} & & & & b^3_{10} & & & & b^3_{20} & & & & b^3_{30} \\
 & & & b^3_{01} & & & & b^3_{11} & & & & b^3_{21} & & & & b^3_{31} \\
 & & & b^3_{02} & & & & b^3_{12} & & & & b^3_{22} & & & & b^3_{32} \\
 & & & b^3_{03} & & & & b^3_{13} & & & & b^3_{23} & & & & b^3_{33} \\
\hline
\end{array} . \tag{44}$$

(44) の $b^j_{ik}$ で示される部分を取り出すために、$NUM(n)$ を $\log n^3$ 回 *stretch* したベクトルと $NUM(n)$ を $\log n^3$ 回 *repeat* したベクトルを作り、これらのベクトルの比較一致演算をしてベクトル $Z_M$ を作る。$Z_M$ でマスクすると、(44) の $b^j_{ik}$ で示される部分以外をすべて 0 にすることができる。(44) を $\log n$ 回 *fold* して、

$$\begin{array}{|cccc|cccc|cccc|cccc|}
\hline
b^0_{00} & b^1_{00} & b^2_{00} & b^3_{00} & b^0_{10} & b^1_{10} & b^2_{10} & b^3_{10} & b^0_{20} & b^1_{20} & b^2_{20} & b^3_{20} & b^0_{30} & b^1_{30} & b^2_{30} & b^3_{30} \\
b^0_{01} & b^1_{01} & b^2_{01} & b^3_{01} & b^0_{11} & b^1_{11} & b^2_{11} & b^3_{11} & b^0_{21} & b^1_{21} & b^2_{21} & b^3_{21} & b^0_{31} & b^1_{31} & b^2_{31} & b^3_{31} \\
b^0_{02} & b^1_{02} & b^2_{02} & b^3_{02} & b^0_{12} & b^1_{12} & b^2_{12} & b^3_{12} & b^0_{22} & b^1_{22} & b^2_{22} & b^3_{22} & b^0_{32} & b^1_{32} & b^2_{32} & b^3_{32} \\
b^0_{03} & b^1_{03} & b^2_{03} & b^3_{03} & b^0_{13} & b^1_{13} & b^2_{13} & b^3_{13} & b^0_{23} & b^1_{23} & b^2_{23} & b^3_{23} & b^0_{33} & b^1_{33} & b^2_{33} & b^3_{33} \\
\hline
\end{array} . \tag{45}$$

を得る。更に $\log n$ 回 *contract* して

$$\begin{aligned}
&(c_{00}, c_{10}, c_{20}, c_{30}, \\
&c_{01}, c_{11}, c_{21}, c_{31}, \\
&c_{02}, c_{12}, c_{22}, c_{32}, \\
&c_{03}, c_{13}, c_{23}, c_{33})
\end{aligned} \tag{46}$$

を得る。ここで、接続行列の性質 $c_{ij} = c_{ji}$ に注意して、例えば

$$c_{10} = c_{01} = b^0_{01} \vee b^1_{01} \vee b^2_{01} \vee b^3_{01} \tag{47}$$

である。∎

今、$MT^{(1)}$ は長さ $n^2$ の接続行列を $mn^n$ 個含んだ $mn^{n+2}$ 長ベクトルであり、各々の行列を自乗しなければならない。そのために長さ $mn^{n+2}$ のベクトル $MT^{(1)}$ を $\log mn^{n+2}$ 回 *repeat*, *stretch* し、$mn^{n+2} \times mn^{n+2}$ のベクトル $MT_r$, $MT_s$ をそれぞれ作り、

$$MT_A := MT_r \wedge MT_s \tag{48}$$

を計算する。次に、$MT_A$ と同じ大きさのベクトル $Z_{DB}$ を構成する。$Z_{DB}$ は図1に示されるように $n^2 \times n^2$ なる大きさで同じ構造の“箱”を $mn^n$ 個持ち、各箱内は (44) と同じように $b^j_{ik}$ のみ1、他は全て0なる構造を持つ。$MT_A$ を $Z_{DB}$ でマスクし、$\log mn^n$ 回 *fold* し、箱が横に $mn^n$ 個並んだ構造に変える。更に $\log n$ 回 *fold* し、$\log n$ 回 *contract* することにより $n$ 行 $(\frac{m}{n} \cdot n^{n+2})$ 列のベクトル $\widetilde{MT^{(2)}}$ を得る。このベクトルを $n$ 列ごとに区切ると、各区分には各行列（接続行列）を自乗した $n \times n$ 行列が含まれていることが分かる。次に図2に示すように、1番目の区分の1行目、2番目の区分の2行目、…、$n$ 番目の区分の $n$ 行目、$n+1$ 番目の区分の1行目、…というように1を立てた（他の部分は0）ベクトル $Z_{MX}$ を作る。$Z_{MX}$ で $\widetilde{MT^{(2)}}$ をマスクし、さらに $\log n$ 回 *fold* することにより、長さ $(\frac{m}{n} \cdot n^{n+2})$ のベクトル $MT^{(2)}$ を得る。これに含まれる各行列は、$MT^{(1)}$ の各行列を自乗したものとなっていることが分かる。ここで $MT^{(1)}$ の各行列の $m$ 回の繰り返しは $MT^{(2)}$ では $\frac{m}{n}$ 回になっていることが分かる。このことは候補ベクトルに含まれる部分ベクトルの $m = n^{\log n}$ 回の繰り返しと深い関係がある。$MT^{(2)}$ に含まれる部分ベクトルに対しても同様に対角要素を調べ、一つでも1が立っていればその行列を取り除く。

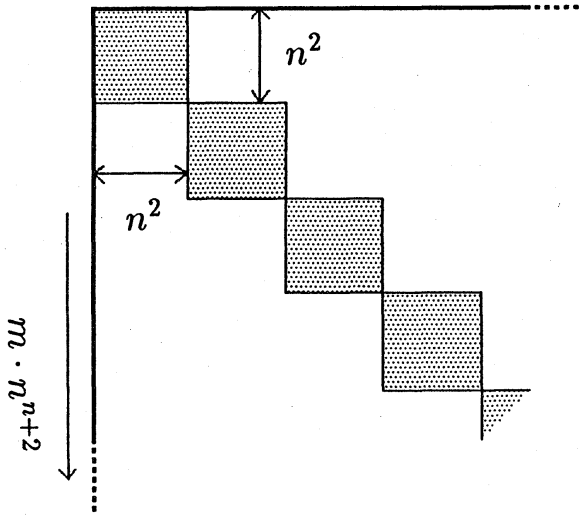


図1．マスクベクトル $Z_{DB}$

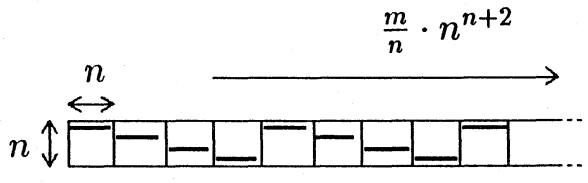


図2．マスクベクトル $Z_{MX}$

$MT^{(2)}$ を作るという問題は $Z_{DB}$, $Z_{MX}$ を如何にして作るかに帰着される。そのために、$NUM'(mn^n)$ を $\log n^2$ 回 *stretch* することにより、

$$M_{DB} = ((1)^{n^2}, (2)^{n^2}, \cdots, (mn^n)^{n^2}) \tag{49}$$

を作り、更に (49) を $\log mn^{n+2}$ 回 *repeat* して $M^r_{DB}$、$\log mn^{n+2}$ 回 *stretch* して $M^s_{DB}$ を作る。これらは、図1の箱の取り出しに用いる。次に、$NUM'(n)$ を $\log m^2 n^{2n+3}$ 回 *repeat* することにより、

$$M_{SB_1} = (1, 2, \cdots, n)^{m^2 \cdot n^{2n+3}} \tag{50}$$

を作り、$NUM'(n)$ を $\log mn^{n+3}$ 回 *stretch* し、更に $\log mn^n$ 回 *repeat* して

$$M_{SB_2} = ((1)^{m\cdot n^{n+3}}, (2)^{m\cdot n^{n+3}}, \cdots, (n)^{m\cdot n^{n+3}})^{mn^n} \tag{51}$$

を作る。$Z_{DB}$ は、 $Z_{DB} := (M^r_{DB} \wedge M^s_{DB}) \wedge (M_{SB_1} \wedge M_{SB_2})$ (52)

で得られる。$Z_{MX}$ は、$NUM'(n)$ を $\log n$ 回 *stretch*、更に $\log mn^n$ 回 *repeat* して、

$$M_{MX_1} = ((1)^n, (2)^n, \cdots, (n)^n)^{mn^n} \tag{53}$$

を作り、$NUM'(n)$ を $\log(\frac{m}{n} \cdot n^{n+2})$ 回 *stretch* して、

$$M_{MX_2} = ((1)^{\frac{m}{n}\cdot n^{n+2}}, (2)^{\frac{m}{n}\cdot n^{n+2}}, \cdots, (n)^{\frac{m}{n}\cdot n^{n+2}}) \tag{54}$$

を作る。$Z_{MX}$ は、$M_{MX_1} \wedge M_{MX_2}$ で得られる。以下同様に、$MT^{(4)}$、$MT^{(8)}$、… を順に構成し、各行列の対角要素を調べ、2 のべきの長さを含む閉路を除く。最後に $MT^{(n)}$ を構成し対角要素が全て 1（長さ $n$ の閉路、つまりハミルトン閉路）である行列が存在するかを調べる。これは、$MT^{(n)}$ を $(DIA(n))^{n^n}$ でマスクし $\log n$ 回 *contract* し、$\log n$ 回 $\wedge$-*contract* する。この時得られるベクトルに一つでも 1 が存在すればハミルトン閉路が存在する。つまり、$\log n^n$ 回 *contract* し、得られた長さ 1 のベクトルの要素が 1 であれば、問題の解は yes である。∎

## 7. むすび

本稿では、具体的問題に対するアルゴリズムについて考察し、充足可能性問題が $O(n)$ 時間、ハミルトン閉路問題が $O(n\log^2 n)$ で解けることを示した。本稿で示したアルゴリズムのうち、ハミルトン閉路問題は、候補ベクトルの作成時間に対し、動作時間の改良の余地が残っている。また、多項式領域完全問題は何乗のオーダーで解けるかについても興味深い問題である。

## 参考文献


[1] A. Bertoni, G. Mauri, and N. Sabadini,"A characterization of the class of functions computable in polynomial time on random access machines," *Proc. 13th ACM Symp. on Theory of Computing*, pp.168-176, 1981.

[2] J. Hartmanis and J. Simon, "On the power of multiplication in random access machines," *Proc. IEEE Symp. on Switching and Automata Theory*, pp.13-23, 1974.

[3] 岩間一雄, "ベクトル機械の一つの標準形," 信学技報, **COMP88**-22, 1988

[4] 岩本, 岩間, "RS 型ベクトル機械," 信学技報, **COMP90**-72, 1990

[5] V. Pratt and L. Stockmeyer, "A characterization of the power of vector machines," *J. Comput. Syst. Sci.*, vol. **12**, pp. 198-221, 1976.

[6] J. Simon, "Division is good," *Proc. 20th IEEE Symp. on Foundations of Computer Science*, pp. 411-420, 1979.