第16卷 第6期 电子科学学刊 Vol. 16, No. 6
1994 年11月 JOURNAL OF ELECTRONICS Nov. 1994

化简开关函数的图论方法

黄汝激
(北京科技大学自动化系 北京 100083)

摘要 本文引入了 n 变量开关函数 $F(x_1,\cdots,x_n)$ 的伴随图 G 和伴随超图 H 的概念，导出了下列方法和算法：(1) 求 F 的所有本原蕴含项的图论方法和分支定界算法 BBAPI；(2) 应用超图理论求 F 的最小和表达式的算法 AMSHT。这些方法简单、直观；既便于手算，也便于用计算机实现；计算效率高于常用的卡诺图法和 Q-M 列表法。

关键词 开关函数，图论，超图理论，分支定界法

1 引 言

开关函数[1]是定义在 $B_2=\{0,1\}$ 上的二元布尔函数。开关函数的表示和化简是开关电路和数字电子线路的分析和设计的基础。常用的表示和化简方法有卡诺图法[2,3]和 Q-M 列表法[3-5]。当开关变量数 $n\leqslant 4$ 时，卡诺图法简单、直观；但当 $n>4$ 时，它突然复杂起来，不便于手算，更不适于用计算机实现。Q-M 列表法可以在计算机上实现，但当 F 的最小项数 $m\geqslant 8$ 时它的计算复杂度 $>O(m^{2.14})$，随着 m 的增大而急剧上升。

开关函数 F 的化简方法通常分为两步[3]：(1) 从 F 的最小项展开式找出 F 的所有本原蕴含项；(2) 从 F 的所有本原蕴含项找出 F 的一个最小覆盖 C_F，C_F 中所有本原蕴含项的和就是 F 的一个最小和表达式。本文将引入 F 的伴随图 G 和伴随超图 H 的概念，并导出求 F 的所有本原蕴含项的图论方法和分支定界算法 BBAPI 以及用超图理论求 F 的最小和表达式的算法 AMSHT。这些方法简单、直观，既便于手算，也便于计算机实现。算法 BBAPI 的计算复杂度为 $O(m^{1.6})$，远小于 Q-M 列表法的 $O(m^{2.14})$。

2 开关函数的拓扑表示和求本原蕴含项的图论方法

开关函数有两种基本展开式：最小项展开式和最大项展开式[3]，二者互为对偶。本文仅考虑最小项展开式的拓扑表示和化简方法。n 个开关变量 $x_1,\cdots,x_n$ 可组成 2^n 个最小项。每个最小项可用一个自然数的 n 位二进制码(或其对应十进制数)表示，称为**最小项代码**，它是把最小项中的原变量和反变量分别用 1 和 0 代替而得到的。因此 2^n 个最小项对应于前 2^n 个自然数，它们构成 2^n 元序列 $B=[0,1,\cdots,2^n-1]=[0\cdots0,$

1993-01-18 收到，1993-08-09 定稿

黄汝激 男，1930 年生，教授，现从事电路、信号与系统，特别是电网络理论、网络图论和超网络超图理论及其应用的研究工作。

$0\cdots1,\cdots,1\cdots1]$，称为 n 位**二进制码序列**. 通常 n 变量开关函数 $F(x_1,\cdots,x_n)$ 的最小项数 $m<2^n$，从 B 中把 F 不包含的最小项的代码去掉后所得 m 元序列称为 $F(x_1,\cdots,x_n)$的**最小项代码序列**，也记作 F.

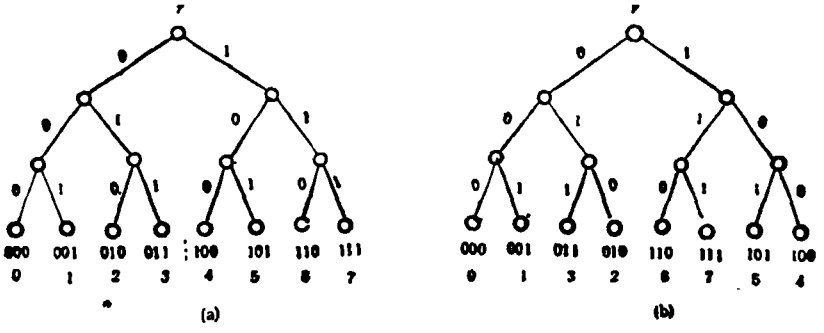


图 1 (a) 二进制码二元树 T_B，(b) 循环码二元树 T_C

n 位二进制码可用 $n+1$ 级**二元树来产生**. 例如三位二进制码可用图 1 (a) 的四级二元树 T_B 来产生，T_B 中从根 r 到每叶的一条路径的边权数组（边权积）就是一个二进制码，这些二进制码从左到右按自然数顺序排列. 如果采用图 1 (b) 的二元树结构，则产生的是循环码，或者说是按循环码顺序排列的二进制码. 把序列 B 的元素改按循环码顺序排列，所得序列记作 C，称为 n **位循环码序列**. 它可由 $n+1$ 级循环码二元树 T_C 来产生. 从图 1 可看出二元树 T_B 和 T_C 的逐级分解为左右两个子树分别对应于序列 B 和 C 的逐级分解为左右两个子序列(称为序列的**二分解**). 考虑到两个最小项可合并的充要条件是它们的代码只有一位不相同，从图 1 可推得下列引理.

引理 1 若把 n 位二进制码序列 B 看作开关函数的最小项代码序列并逐级进行二分解，则第 i 级二分解 $(1\leqslant i\leqslant n)$ 所得两个 2^{n-i} 元子序列的每对**同序号**二进制码可以合并，合并后消去了第 i 位码(即第 i 位变为空位，记作 θ).

引理 2 若把 n 位循环码序列 C 看作开关函数的最小项代码序列并逐级进行二分解，则第 i 级二分解 $(1\leqslant i\leqslant n)$ 所得两个 2^{n-i} 元子序列中对于其分解界线对称(称为 i **级对称**)的每对二进制码可以合并，合并后消去第 i 位码(记作 θ).

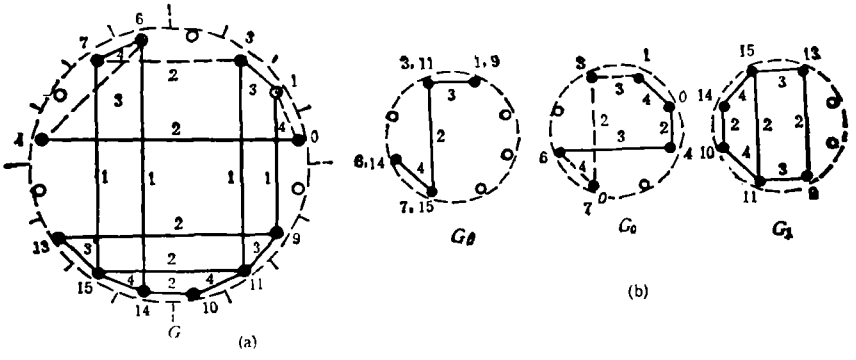
开关函数可以用拓扑图表示如下：先把循环码序列 C 的 2^n 个元素用均匀分布在一个圆周上的 2^n 个空心点表示；对应于开关函数的最小项代码序列 F 的那些空心点改用实心点(简称点)表示，所有实心点的集合称为**点集**，记作 V；满足 i 级对称 $(1\leqslant i\leqslant n)$ 的每对实心点用一条权为 i 的边连接起来，所有边的集合称为**边集**，记作 L；由点集 V 和边 L 集构成的图 $G=(V,L)$ 称为开关函数 F 的**伴随图**(简称 F 的图).

图 G 中含有许多称为**布尔格**[1]的子图. 每个实心点是一个 **0 维格** L^0. 两个 0 维格若被一条权为 w_1 的边连接起来则构成一个边权为 w_1 的 **1 维格** L^1. 依此类推，两个 $r-1$ 维格若被 2^{r-1} 条权为 w_r 的边互联起来，使得每点度数为 r，则形成一个边权集为 $\{w_1,\cdots,w_r\}$ 的 r **维格** L^r. G 中 r 维格对应于 F 中蕴含项 $(0\leqslant r\leqslant n)$，它们的二进制代码可由格中任一点的二进制码把第 $w_1,\cdots,w_r$ 位改为空位 θ 得到. G 中维数不

能再扩大的 r 维格称为 G 的**本原 r 维格**。它对应于 F 的本原蕴含项。根据上述分析可得

定理 1　开关函数 F 的伴随图 G 中子图 L^r 成为本原 r 维格的充分必要条件是：(1) L^r 含 2^r 个点和 $r2^{r-1}$ 条边；(2) 每点关联 r 条边，而且边权集都相同，都是 $\{w_1,\cdots,w_r\}\subset\{1,\cdots,n\}$；(3) 不存在 s 维格 $L^s\supset L^r, s>r$。

根据定理 1 可从图 G 中找出它的所有本原 r 维格 L^r 及其边权集 $\{w_1,\cdots,w_r\}(0\leqslant r\leqslant n)$，把每个 L^r 中任一点的二进制码内第 $w_1,\cdots,w_r$ 位改为空位 θ，就是 L^r 及其对应本原蕴含项的代码，从而可求得 F 的所有本原蕴含项。



——最小覆盖格，- - - 非最小覆盖格　　——本原格，- - - 非本原格

图 2　(a) 开关函数 F 的图 $G=(V,L)$　　(b) G 的一级分解图 $\{G_\theta,G_0,G_1\}$

例 1　四变量开关函数

$$\begin{aligned}F(x_1,x_2,x_3,x_4)&=\bar{x}_1\bar{x}_2\bar{x}_3\bar{x}_4+\bar{x}_1\bar{x}_2\bar{x}_3x_4+\bar{x}_1\bar{x}_2x_3x_4+\bar{x}_1x_2\bar{x}_3\bar{x}_4+\bar{x}_1x_2x_3\bar{x}_4\\&\quad+\bar{x}_1x_2x_3x_4+x_1\bar{x}_2\bar{x}_3x_4+x_1\bar{x}_2x_3\bar{x}_4+x_1\bar{x}_2x_3x_4+x_1x_2\bar{x}_3x_4\\&\quad+x_1x_2x_3\bar{x}_4+x_1x_2x_3x_4\\&=\Sigma(0,1,3,4,6,7,9,10,11,13,14,15)\end{aligned}$$

的伴随图 $G=(V,L)$ 如图 2 (a) 所示。点集 $V=0$ 维格集 $L^0=\{0,1,3,4,6,7,9,10,11,13,14,15\}$，边集 $L=1$ 维格集 $L^1=\{(1,9),(3,11),(6,14),(7,15);(0,4),(3,7),(9,13),(11,15),(10,14);(1,3),(6,4),(13,15),(9,11);(0,1),(6,7),(15,14),(10,11)\}$。边(1,9)和(3,11)通过 3 级对称边(1,3)和(9,11)互联成 2 维格(1,9,3,11)，它含有 $2^2=4$ 个点和 $2\cdot 2^{2-1}=4$ 条边，每点关联两条边，边权集都是 $\{1,3\}\subset\{1,2,3,4\}$。2 维格集 $L^2=\{(3,11,7,15),(1,9,3,11),(6,14,7,15),(9,13,11,15),(11,15,10,14)\}$，其中都是本原 2 维格。本原 1 维格有三个：(0,4)，(0,1)，(6,4)。2 维和 1 维本原格代码各为 $\theta\theta11,\theta0\theta1,,\theta11\theta,1\theta\theta1,1\theta1\theta$ 和 $0\theta00,000\theta,01\theta0$。因此求得 F 的所有本原蕴含项为 $x_3x_4,\bar{x}_2x_4,x_2x_3,x_1x_4,x_1x_3,\bar{x}_1\bar{x}_3\bar{x}_4,\bar{x}_1\bar{x}_2\bar{x}_3$ 和 $\bar{x}_1x_2\bar{x}_4$。

如果变量数 n 和最小项数 m 较大，则图 G 较复杂，不易看清。为了简化图形，可把 G 分解成三个子图 G_θ,G_0 和 G_1，称为 F 和 G 的**一级分解图**，记作 $\{G_\theta,G_0,G_1\}$。G_θ 是 G 中所有权为 1 的边的关联点集上的导出子图[6]，不过把每条权为 1 的边(即代码第一位为 θ 的 1 维格)收缩成一点，下半圆收缩到上半圆，再把所得上半圆内的图展开(拓扑变形)

成一个圆内的图. G_θ 中所有实心点构成的序列是代码第一位为 θ 的一维格序列，记作 F_θ. G_0 和 G_1 是从 G 中移去所有权为 1 的边后得到的上半圆和下半圆内的子图，不过分别把它们展开成一个圆内的图. G_0 和 G_1 中所有实心点构成的序列是代码第一位分别为 0 和 1 的 0 维格序列，各记作 F_0 和 F_1. 例如图 2(a) 的图 G 的一级分解图如图 2 (b) 所示. 显然，$\{G_\theta, G_0, G_1\}$ 含有 G 的所有本原格. 不过从 $\{G_\theta, G_0, G_1\}$ 求 G 的所有本原格时，要注意去掉非本原格. 例如图 2 中 G_0 的一维格(1,3)(3,7)和(6,7)是非本原格，因为它们各含在二维本原格(1,9,3,11),(3,11,7,15)和(6,14,7,15)中. 如果子图 G_θ, G_0 或 G_1 仍较复杂，可以逐级继续分解下去，形成求 F 的所有本原蕴含项的**分解图法**. 分解过程中代码**前部**(即前几位组成的数组)为 h 的所有格构成的序列称为**前部为 h 的格序列**，记作 F_h. G 中以 F_h 为点集的导出子图记作 G_h.

上述图论方法是根据引理 2 和定理 1 采用循环码序列 C 构成的，可称为**图论方法 1**. 类似地，根据引理 1 和定理 1 采用二进制码序列 B 代替 C 可构成**图论方法 2**.

3 求所有本原蕴含项的分支定界算法

引入下列记号：F——开关函数 $F(x_1, \cdots, x_n)$ 的最小项代码序列，n——F 的变量数，h——n 位二进制码的前部，F_h，M_h——前部为 h 的格代码序列及其标记序列，m——F_h 的元素数，Sh, SF_h, SM_h, Sm——存放 h，F_h，M_h 和 m 的堆栈，t——栈指针，a, b, c, d——存放二进制代码的数组，$h\theta c$——由 h, θ 和 c 串接成的代码，PL——本原格代码序列，L——按格维数由高到低排序的本原格代码序列，E——按格维数由高到低排序的本原格点集序列，e——L 的元素数.

根据图论方法 2 的思路可设计一个便于在计算机上实现的**求 F 的所有本原蕴含项的分支定界算法 BBAPI**，它的步骤如下：

(1) 输入 n 和 F.

(2) 初始化：$h \leftarrow \phi, F_h \leftarrow F, M_h \leftarrow \phi, m \leftarrow |F|, PL \leftarrow \phi, L \leftarrow \phi, E \leftarrow \phi, t \leftarrow 0, p \leftarrow 0$.

(3) 若 $m > 1$，转(4)；若 $m = 1$，转(7)；若 $m = 0$，转(9).

(4) 分解 F_h 为两个子序列 F_{h0} 和 F_{h1}：

(a) $j \leftarrow 0, k \leftarrow 0, l \leftarrow 0$.

(b) 对于 $i = 1$ 到 m 进行 $a \leftarrow F_h(i)$，若 $a(|h| + 1) = 0$，则 $j \leftarrow j + 1, F_{h0}(j) \leftarrow F_h(i), M_{h0}(j) \leftarrow M_h(i)$，否则 $k \leftarrow k + 1, F_{h1}(k) \leftarrow F_h(i), M_{h1}(k) \leftarrow M_h(i)$.

(c) $m_0 \leftarrow j, m_1 \leftarrow k$.

(5) 若 $m_0 = 0$ 则 $h \leftarrow h1, F_h \leftarrow F_{h1}, M_h \leftarrow M_{h1}, m \leftarrow m_1$，转 (3). 若 $m_1 = 0$ 则 $h \leftarrow h0, F_h \leftarrow F_{h0}, M_h \leftarrow M_{h0}, m \leftarrow m_0$，转 (3).

(6) 若 $m_0 > 0$ 与 $m_1 > 0$ 则产生 $F_{h\theta}$：

(a) $j \leftarrow 0, k \leftarrow 0, l \leftarrow 0$.

(b) $j \leftarrow j + 1$，$k \leftarrow k + 1$；若 $j \leqslant m_0$ 和 $k \leqslant m_1$ 则 $a \leftarrow F_{h0}(j), b \leftarrow F_{h1}(k)$，对于 $i = |h| + 2$ 到 n 进行 $c(i - |h| - 1) \leftarrow a(i)$，$d(i - |h| - 1) \leftarrow b(i)$，转 (c)，

否则转 (f).

(c) 若 $c=d$ 则 $l\leftarrow l+1, F_{h\theta}(l)\leftarrow h\theta c, M_{h\theta}(l)\leftarrow 0, M_{h0}(j)\leftarrow 1, M_{h1}(k)\leftarrow 1$, 转 (b).

(d) 若 $c<d$ 则 $j\leftarrow j+1$; 若 $j\leqslant m_0$ 则 $a\leftarrow F_{h0}(j)$, 对于 $i=|h|+2$ 到 n 进行 $c(i-|h|-1)\leftarrow a(i)$, 转 (c), 否则转 (f).

(e) 若 $c>d$ 则 $k\leftarrow k+1$; 若 $k\leqslant m_1$ 则 $b\leftarrow F_{h1}(k)$, 对于 $i=|h|+2$ 到 n 进行 $d(i-|h|-1)\leftarrow b(i)$, 转 (c), 否则转 (f).

(f) $m_\theta\leftarrow l$.

(g) 若 $m_\theta=0$ 则(1)若 $m_1>1$ 或 $m_1=1$ 和 $M_{h1}(1)=0$ 则 $t\leftarrow t+1, Sh(t)\leftarrow h1, SF_h(t)\leftarrow F_{h1}, SM_h(t)\leftarrow M_{h1}, Sm(t)\leftarrow m_1$;(2) $h\leftarrow h0, F_h\leftarrow F_{h0}, M_h\leftarrow M_{h0}, m\leftarrow m_0$, 转(3).

(h) 若 $m_\theta>0$ 则(1)若 $m_1>1$ 则 $t\leftarrow t+1, Sh(t)\leftarrow h1, SF_h(t)\leftarrow F_{h1}, SM_h(t)\leftarrow M_{h1}, Sm(t)\leftarrow m_1$;(2)若 $m_0>1$ 则 $t\leftarrow t+1, Sh(t)\leftarrow h0, SF_h(t)\leftarrow F_{h0}, SM_h(t)\leftarrow M_{h0}, Sm(t)\leftarrow m_0$;(3) $h\leftarrow h\theta, F_h\leftarrow F_{h\theta}, M_h\leftarrow M_{h\theta}, m\leftarrow m_\theta$, 转(3).

(7) 若 $M_h(1)=1$ 则转(9);否则 $a\leftarrow F_h(1)$, 对于 $i=1$ 到 p 进行: $b\leftarrow PL(i)$, 若 $a\subset b$ [即 $a(j)\subseteq b(j), j=1,\cdots,n$; 这里关系$\subset$定义为 $0\subset\theta$ 和 $1\subset\theta$] 则转(9).

(8) $p\leftarrow p+1, PL(p)\leftarrow a$.

(9) 若 $t>0$ 则出栈: $h\leftarrow Sh(t), F_h\leftarrow SF_h(t), M_h\leftarrow SM_h(t), m\leftarrow Sm(t), t\leftarrow t-1$, 转(3).

(10) $e\leftarrow p$, 把 PL 中所有本原格代码按所含 θ 数由大到小重排序后送入 L, 对于 $i=1$ 到 e 进行: $E(i)\leftarrow L(i)$ 的对应点集(即把格代码 $L(i)$ 中每个 θ 换成 0 和 1 得到的 0 维格代码集). 结束.

计算时间复杂度分析　考虑 F 的最小项数 $m\approx 2^r$ 的情况, 算法的计算量决定于 F 的图 G 中所含格的总数 n_L, 当 $m=2^r$ 且 G 为 r 维格时, n_L 最大, 计算量最大. 算法 BBAPI 的计算量主要决定于步骤(4)和(6). 当 G 为 r 维格时, 步骤(4)和(6)处理的格总数都是

$$n_B=m+\left(\frac{3}{2}\right)m+\cdots+\left(\frac{3}{2}\right)^r m=2m\left[\left(\frac{3}{2}\right)^{r+1}-1\right]$$

$$=3^{r+1}-2^{r+1}\approx 3(2^{1.6r})=3m^{1.6}, \tag{1}$$

因此算法 BBAPI 的计算时间复杂度为 $O(m^{1.6})$. Q-M 列表法是把 G 中所有 i 维格按格代码中 1 的个数 j 由小到大分组, 然后把 j 号组的每个 i 维格与 $j+1$ 号组的每个 i 维格进行比较来产生 $i+1$ 维格. 当 G 为 r 维格时, j 号组的 i 维格数为 $C_r^{i+j}C_{i+j}^i$, 它的计算(比较)总次数为

$$n_Q=\sum_{i=0}^{r-1}\sum_{j=0}^{r-i-1}C_r^{i+j}C_{i+j}^i\cdot C_r^{i+j+1}C_{i+j+1}^i=\sum_{i=0}^{r-1}(C_r^i)^2\sum_{j=0}^{r-i-1}C_{r-i}^jC_{r-i}^{j+1}$$

$$=\sum_{i=0}^{r-1}(C_r^i)^2C_{2(r-i)}^{r-i-1} \tag{2}$$

设 $r \geqslant 8$ 且为偶数，令 $I(x)$ 表示不小于 x 的最小整数，

$$y_k = I\left(\frac{k-2}{k-1}\ \frac{r}{2}\right),\ \Delta y_k = y_{k+1} - y_k \approx I\left(\frac{r}{2k(k-1)}\right),$$

则 $\prod_{j=y_k}^{y_{k+1}} \left(\frac{r-j}{\frac{r}{2}-j}\right) \geqslant k^{\Delta y_k}$，从而

$$C_r^{r/2} = \prod_{j=0}^{\frac{r}{2}-1} \left(\frac{r-j}{\frac{r}{2}-j}\right) = \prod_{j=y_2}^{y_3} \left(\frac{r-j}{\frac{r}{2}-j}\right) \prod_{j=y_3}^{y_4} \left(\frac{r-j}{\frac{r}{2}-j}\right) \prod_{j=y_4}^{r/2} \left(\frac{r-j}{\frac{r}{2}-j}\right)$$

$$> 2^{\Delta y_2} 3^{\Delta y_3} 4^{(r/2-y_4)} \geqslant 2^{r/4} 3^{r/12} 4^{r/6} = m^{0.714}, \tag{3}$$

考虑(2)式右边的最大项 $(C_r^{r/2})^2 C_r^{r/2-1} \approx (C_r^{r/2})^3 > m^{2.14}$，因此 Q-M 列表法的计算时间复杂度超过 $O(m^{2.14}) \gg$ BBAPI 的 $O(m^{1.6})$.

例 2 应用算法 BBAPI 求 $F = [0011, 0100, 0111, 1001, 1011, 1111]$ 的所有本原蕴含项的过程(状态空间树)如图 3 所示.

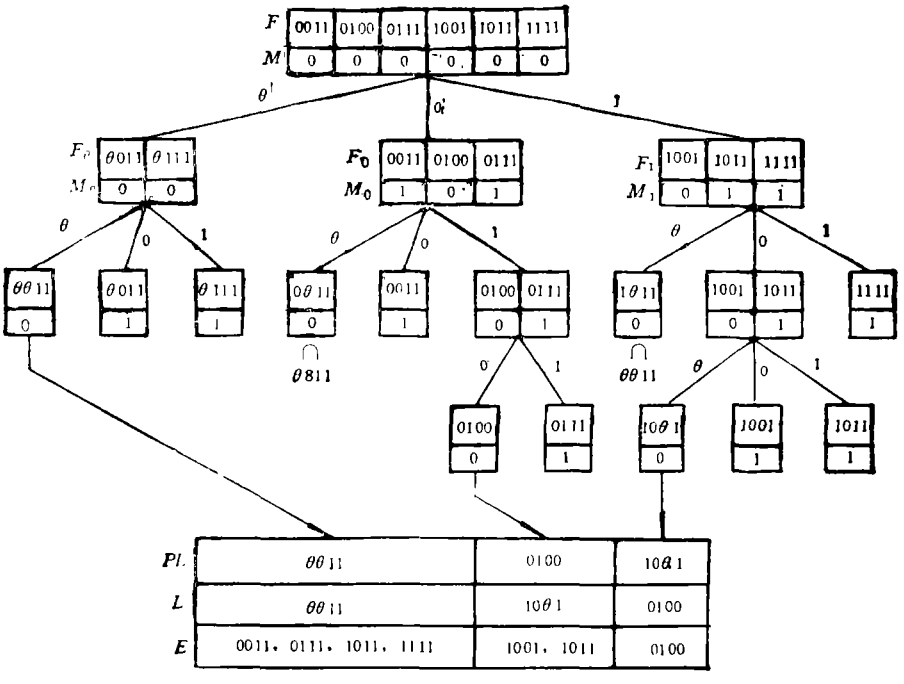


图 3 例 2 中算法 BBAPI 的状态空间树
(图中的 θ811 应为 θθ11)

4 应用超图理论求最小和表达式

考虑一个开关函数 F 及其图 G，能包含 G 中所有点的一些本原格构成的集合称为 G 的一个**格覆盖**. G 的所有格覆盖中基数(即所含本原格数)最小且格代码中 0，1 总数最小的格覆盖称为 **G 的最小格覆盖** C_G. 由 C_G 中本原格的对应本原蕴含项构成的集合称

为 F 的**最小覆盖** C_F。C_F 中所有本原蕴含项的和就是 F 的**最小和表达式**。因为该表达式的项数最少，而且出现的开关变量总数也最少。

当图 G 较简单时，可由观察直接找到 G 的最小格覆盖 C_G。当 G 较复杂时，可应用超图理论求 C_G。如果把 G 中每个本原格的点集 $E_i(i=1,\cdots,e)$ 看成一条超边，则 G 的点集 $V=\{v_1,v_2,\cdots,v_m\}$ 和超边集 $E=\{E_1,E_2,\cdots,E_e\}$ 构成一个**超图**[7] $H=(V,E)$，称为开关函数 F 或图 G 的**伴随超图**（简称 F 或 G 的超图）。在 H 中若 E_i 含有 v_j 则称 E_i **关联**（或**覆盖**）v_j 或 v_j 关联 E_i。点 v_j 所关联的所有超边的集合称作 v_j 的**关联超边集**，记作 V_j。例如，例 1 中的开关函数 F 的超图 H 如图 4 所示，其中 $v_2=1$ 的关联超边集 $V_2=\{E_2,E_7\}$。如果 E 的一个子集 P 能关联 H 中所有点，则称 P 为 H 的一个**超边覆盖**。H 的所有超边覆盖中基数（即所含超边数）最小且所含超边的点数总和最大的超边覆盖称为 H 的**最小超边覆盖**，记作 C_H。因为本原格对应于超边，所以有

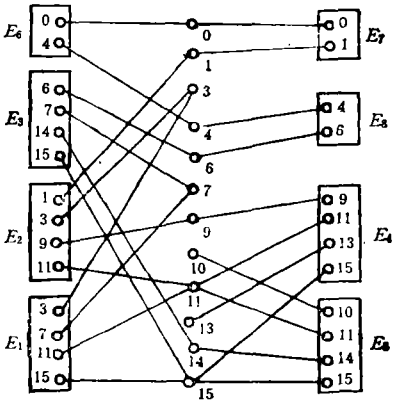


图 4　例 1 中开关函数 F 的超图 H（图中实线为引线，表示点与超边的关联关系）

定理 2　求图 G 的最小格覆盖 C_G 的问题等价于求 G 的伴随超图 H 的最小超边覆盖 C_H 的问题。

引入记号：H_m——被处理超图，H_a,H_b——H_m 的部分超图，C——似最小超边覆盖（即在含有或不含某些超边的条件下的最小超边覆盖），D——似最小超边覆盖集，p,q——C 和 D 的计数器，SH_a,SC,Sp——H_a,C 和 P 的堆栈，t——栈指针。

根据定理 2 可导出**应用超图理论求开关函数 F 的一个最小和表达式的算法 AMSHT** 如下：

（1）输入 F 的最小项代码集 V 和本原蕴含项集（即算法 BBAPI 的输出本原格点集序列）E，以 V 为点集、E 为超边集构成超图 $H=(V,E)$。

（2）初始化：$H_m\leftarrow H,p\leftarrow 0,q\leftarrow 0,t\leftarrow 0,k\leftarrow 0$。

（3）在 $H_m=(V,E)$ 中，检查 V 中每个点 v_i：若 $\deg v_i\triangleq|V_i|=1,V_i=\{E_i\}$，则 $p\leftarrow p+1,C(p)\leftarrow E_i,V\leftarrow V-E_i,E\leftarrow E-\{E_i\},k\leftarrow 1$。（$E_i$ 称为**实质超边**，必有 $E_i\subset C$，否则 C 不能覆盖 v_i）

（4）检查 E 中每条超边 E_i：若存在 E_j 使 $E_i\subseteq E_j$（称 E_i 为**劣势超边**）且 E_i 的代码 L_i 中 0，1 总数 $\geqslant L_j$ 中 0，1 总数，则 $E\leftarrow E-\{E_i\},k\leftarrow 1$（劣势超边 E_i 可以去掉，因为用 E_j 比用 E_i 可能更有利于构成最小超边覆盖）。

（5）检查 V 中每个点 v_i：若存在 v_j 使 $V_i\supseteq V_j$（称 v_j 为**优势点**），则 $V\leftarrow V-\{v_j\},k\leftarrow 1$（优势点 v_j 可以去掉，因为当 v_i 被某超边覆盖时 v_j 一定也被该超边覆盖）。

（6）若 $k=0$，则转（7）；否则 $k\leftarrow 0$，转（3）[即重复进行步骤（3）到（5）直到没有实质超边、劣势超边和优势点（$k=0$）时转（7）]。

(7) 若 $H_w = \phi$，转(9)；否则 (H_w 含**超回路**[7]) 转(8)。

(8) 用分支定界法找出 H_w 的**似最小超边覆盖集** D：选一基数最大的超边 E_i (若有几个则选其中下标最小者，这样可使下面得到的 H_a 和 H_b 较简单)，分两支进行：

(a) $E_a \leftarrow E - \{E_i\}, H_a \leftarrow (V, E_a); t \leftarrow t+1, SH_a(t) \leftarrow H_a, SC(t) \leftarrow C, SP(t) \leftarrow p$。

(b) $p \leftarrow p+1, C(p) \leftarrow E_i; V_b \leftarrow V - E_i, E_b \leftarrow E - \{E_i\}, H_b \leftarrow (V_b, E_b); H_w \leftarrow H_b$，转(3)。

(9) $q \leftarrow q+1, D(q) \leftarrow C$。

(10) 若 $t > 0$ 则出栈：$H_w \leftarrow SH_a(t), C \leftarrow SC(t), p \leftarrow Sp(t), k \leftarrow 0$，转(3)。

(11) 输出：从 D 中选出一个**最小超边覆盖** $C_H = \{E_i\}$，求出 C_H 的对应本原格代码集 (即 G 的**最小格覆盖**) C_G 和 F 的**最小覆盖** $C_F = \{F_i\}$，输出 F 的**最小和表达式** $F = \Sigma F_i$。

注意：(1) 人工计算时，可在超图 H 上直接应用算法 AMSHT 进行，比较简单、直观。(2) 用计算机计算时，要根据算法 AMSHT 编写程序，这时超图 H 可用它的**关联矩阵** A 来表示。$A = [a_{ij}]$ 的行 i 和列 j 分别表示 H 的超边 E_i 和点 v_j，若 E_i 关联 v_j 则 $a_{ij} = 1$，否则 $a_{ij} = 0$。

例 3 **例** 1 中开关函数 F 的超图 $H = (V, E)$ 如图 4 所示，应用算法 AMSHT 求 F 的最小和表达式，输出结果如下：

$D = \{\{E_4, E_5, E_1, E_7, E_8\}, \{E_4, E_5, E_2, E_3, E_6\}\}$,

$C_H = \{E_4, E_5, E_2, E_3, E_6\}$,

$C_G = \{1\theta\theta1, 1\theta1\theta, \theta0\theta1, \theta11\theta, 0\theta00\}$,

$C_F = \{x_1x_4, x_1x_3, \bar{x}_2x_4, x_2x_3, \bar{x}_1\bar{x}_3\bar{x}_4\}$,

$F = x_1x_4 + x_1x_3 + \bar{x}_2x_4 + x_2x_3 + \bar{x}_1\bar{x}_3\bar{x}_4$。

观察图 2 和图 4 可以验证所得结果的正确性。

参 考 文 献

[1] Samuel C L. Modern Switching Theory and Digital Design. Englewood Cliffs, N. J.: Prentice-Hall, Inc., 1978.

[2] Karnaugh M. Communications and Electronics, 1953, (9): 593—599.

[3] Muroga S. Logic Design and Switching Theory, New York: John Wiley & Sons, 1979, Chapter 3—4.

[4] Quine W V. American Mathematics Monthly, 1952, 59(8): 521—531.

[5] McCluskey E J. Bell Systems Technical Journal, 1956, 35(6): 1417—1444.

[6] Swamy M N S, 等著，左哲主译. 图、网络与算法. 北京：高等教育出版社，1988年，第一章.

[7] 黄汝激. 电子科学学刊，1987，9(3): 244—255.

GRAPH THEORY METHODS FOR SIMPLIFICATION OF SWITCHING FUNCTIONS

Huang Ruji

(*Beijing University of Science and Technology, Beijing* 100083)

Abstract The concepts of associated graph G and associated hypergraph H are introduced for a switching function $F(x_1,\cdots,x_n)$ with n variables. Then, the following methods and algorithms are derived: (1) The graph theory methods and branch-and-bound algorithm BBAPI for finding all prime implicants of F. (2) Algorithm AMSHT for finding a minimal sum for F by hypergraph theory. These methods are simple and intuitive. They are convenient not only for computation by hand but also for realization by a computer. Their computation efficency are higher than that of Karnaugh map method and Q-M tabular method which are customarily used.

Key words Switching function, Graph theory, Hypergraph theory, Branch-and-bound method