基于 OBDD 的有向图的存储与操作研究*)

杨志飞　古天龙
（桂林电子科技大学计算机系　桂林 541004）

摘　要　本文讨论了一种基于 OBDD 的有向图的存储结构，给出了基于 OBDD 的有向图的操作方法及搜索算法。实验结果表明，该存储结构与传统的邻接表的存储结构相比，在处理大规模的有向图时，具有较高的存储效率。

关键词　数据结构，存储结构，有序二叉决策图(OBDD)

A Novel OBDD Technique to Store and Manipulate Directed Graphs

YNAG Zhi-Fei　GU Tian-Long
(School of Computer Science, Guilin University of Electronic Technology, Guilin 541004)

Abstract　The novel OBDD based storage structure of directed graphs is proposed, and the operations and algorithms are presented. The experimental results show that the storage structure works well, especially for large size directed graphs.

Keywords　Data structure, Storage structure, Ordered binary decision diagram (OBDD)

图是一种具有广泛应用的重要数据结构和模型。它的主要存储结构有：顺序存储和链式存储。其中顺序存储会给图的各种操作在算法实现上带来相当的难度，甚至无法实现[1]，因而图的存储结构常使用链式存储，即：邻接表、十字链表、邻接多重表等。邻接表是有向图典型存储方式。

有序二叉决策图(OBDD- ordered binary decision diagram)是一种用于隐式表示布尔函数的数据结构[2,3]，是迄今为止最为有效的符号技术之一。本文讨论了有向图的有序二叉决策图存储结构。它的设计思想不同于传统的链式存储结构。该存储方式将图的顶点进行编码、图的边转化为二元布尔关系，从而将整个有向图转化为布尔函数并用 OBDD 的形式表示。在这种表示形式下，图的所有的操作都可以转化为布尔函数的运算，从而可以通过基于 OBDD 的图算法进行操作。它的优点在于能够以较小的空间存储较大规模的有向图。

1　有向图的 OBDD 表示

有序二叉决策图 OBDD 是用于表示基于变量 $x_1, x_2, \cdots, x_n$ 的一簇布尔函数 $f_i:\{0,1\}^n \to \{0,1\}$ 的一个有向无环图 $G(V \cup T, E)$，它满足：图中每个结点对应唯一的函数 f_i；没有射出边的结点称为终结点，且仅有两个终结点 0 和 1，终结点集合记为 T；除终结点外的其它结点称作内结点，记内结点的集合为 V。对于 $\forall v \in V$，均由变量名 $var(v)$ 标识，且有两条射出边。v 取值 0 后的射出边称为 0-边，v 取值 1 后的射出边称为 1-边。所有边的集合记为 E。0-边和 1-边所指向的结点分别记为 $low(v)$ 和 $high(v)$；在 *OBDD* 的有向路径上，每个变量至多出现一次；给定变量序 $\pi: x_1 < x_2 < \cdots < x_n$ 时，在每条有向路径上都必须以该变量次序出现在图形表示中，一般用方框表示终结点，用圆圈表示内节点。通常，假设边的方向朝下，0-边用虚线表示，1-边用实线表示。

例如：布尔函数 $F = x_1' x_2' y_1 y_2' + x_1' x_2' y_1' y_2 + x_1' x_2 y_1 y_2 + x_1 x_2 y_1 y_2'$，在变量序 $x_1 < y_1 < x_2 < y_2$ 下的 *OBDD* 表示如图 1 所示。

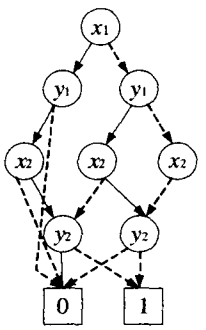

图 1　布尔函数的 OBDD 表示

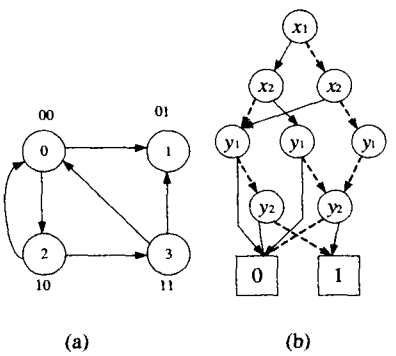

图 2　有向图的 OBDD 表示

对于有向图 $G=(V,E)$，可以对结点进行 n 位二进制编码：$n=\lceil \log_2 |V| \rceil$。有向边可理解为结点之间的关系，顶点 v_0 到顶点 v_1 的边，可用特征函数 $E(v_0, v_1)=1$ 来描述。设 $X=(x_1, \cdots, x_n)$，$Y=(y_1, \cdots, y_n)$ 是图中结点的二进制编码向量，则结点 X 到结点 Y 的边的特征函数表示为：

*)基金项目：国家自然科学基金项目(60563005)；教育部留学归国人员基金项目。**古天龙**　CCF 会员。

$E(X,Y):\{0,1\}^n\times\{0,1\}^n\rightarrow\{0,1\}$

有向图G的边集E可以看作是布尔关系的集合，而布尔关系的集合可以用布尔函数表示，因而可以用OBDD来存储有向图。

例如，图2(a)所示有向图G的OBDD存储结构示于图2(b)。

CUDD软件包是美国Colorado大学开发的OBDD软件包[4]，它提供了丰富的BDD操作函数。在该软件包中OBDD的结点结构如下：

```
typedef struct DdChildren {
  struct DdNode * T;/ * 结点的1-边 * /
  struct DdNode * E;/ * 结点的0-边 * /
}DdChildren;
struct DdNode {
       DdHalfWord index;/ * 索引值 * /
       DdHalfWord ref;/ * 引用计数 * /
       DdNode * next;/ * 指向唯一表的下一个指针 * /
union{
       CUDD_VALUE_TYPE value;
         / * 常量结点 * /
DdChildren kids;
           / * 内结点 * /
         }type;
};
```

2 基于OBDD的有向图的操作和算法

下面给出OBDD存储结构下，有向图的一些操作和宽度优先搜索算法。

2.1 有向图的操作

求顶点的度 给定有向图的OBDD表示$E(X,Y)$和一个顶点的编码$X=(x_1,\cdots,x_n)$，则该顶点的出度可以通过计算式$\exists X(E(X,Y))$的成真赋值的个数得到；把该顶点的编码换为$Y=(y_1,\cdots,y_n)$，则该顶点的入度可通过计算式$\exists Y(E(X,Y))$的成真值赋值的个数得到。

增加边 设需要增加的边的集合为E_{add}，建立该集合特征函数的OBDD表示$E_{add}(X,Y)$，则可以以OBDD的方式计算下式得到：$E(X,Y)+E_{add}(X,Y)$。该操作通过OBDD的APPLY操作来完成。

删除边 设需要删除边的集合为E_{del}，建立该集合特征函数的OBDD表示$E_{del}(X,Y)$，则可以以OBDD的方式计算下式得到：$E(X,Y)\cdot(E_{add}(X,Y))'$。

查找边 设该边的起始结点$X=(x_0,\cdots,x_{n-1})$，终结点$Y=(y_0,\cdots,y_{n-1})$，由结点X和X编码的赋值检测$E(X,X)$中是否有一条路径到达1结点。如果有，则表示图中存在该边，否则，该边不存在。

2.2 宽度优先搜索算法

设v_0、v_g分别为有向图$G=(V,E)$的起始顶点和目标顶点。设S_i为从初始顶点开始进行第i步搜索而得到的顶点的集合，初始化为$S_0=\{v_0\}$，目标顶点的集合为$S_g=\{v_g\}$。

在该算法中，给定有向图的OBDD表示$E(X,Y)$和顶点集合S_{i-1}的OBDD表示$S_{i-1}(x)$，可以由下面的计算得到$S_i(x)$(即第i步搜索到的顶点的集合)：

$$S_i(X)=(\exists X(S_{i-1}(X)\cdot E(X,Y))[Y/X]$$

上述运算首先通过OBDD的关系积运算$S_{i-1}(X)\cdot E(X,Y)$得到起始结点$X\in S_{i-1}(X)$的边的集合$S_{i-1}(X)$，再通过OBDD的存在量化操作得到边集$S_{i-1}(X)$中所有边的终结点的集合$S_i(Y)$。由于该终结点的集合由布尔变量向量Y标识，因此需要进行一个简单的布尔变量替换操作$[Y/X]$，就可以得到由$S_{i-1}(X)$中的结点所能够到达的所有结点的集合。通过迭代调用，便可以实现图的宽度优先搜索算法，伪码如下：

```
procedure Breadth_first-Search
    Open ←S0(X)
    do
        Succ←(∃Y((Open(X) · E(X,Y))[Y/X]
        Open←Succ
        while (Open(X) · Sg(X)≡0)
```

在上述算法中，使用了两个OBDD变量Open和Succ，Open表示待扩展顶点的集合，Succ是Open的后继顶点的集合。由于算法实际上对当前结点集中的每个结点均进行了扩展，因此当算法结束的时候，算法循环迭代的次数就是最优路径的长度。

3 仿真实验

我们利用CUDD软件包在运行平台Windows2000，P4 1500MHZ CPU，128MB RAM下，对随机产生的不同密度的有向图进行了实验。变量按交叉排序，即$x_1<y_1<x_2<y_2<\cdots$。实验结果如表1所示。

表1 实验结果

序号	有向图		OBDD	邻接表
	顶点数	边数	结点数	结点数
1	1414	200000	98995	201414
2	1000	200000	73635	201000
3	817	200000	65997	200817
4	708	200000	55522	200708
5	633	200000	46280	200633
6	577	200000	39806	200577
7	535	200000	31921	200535
8	500	200000	21953	200500
9	471	200000	15229	200471
10	800	629200	113	630000
11	448	200256	83	200704
12	1000	499500	101	500500

邻接表中的一个顶点结点和边结点均占8个字节，在CUDD包中一个OBDD结点占16个字节，因此一个OBDD结点相当于2个邻接表的结点。通过比较2V1/V2(V1为OBDD存储所需要的结点数目，V2为邻接表存储所需要的结点数目)可以看出：在边数不变的情况下，随着顶点数的减少，OBDD比邻接表节省的存储空间越来越多。

图3给出了具有100个结点、不同边数的随机图的存储效率比较。当边数为7000时，OBDD所需的存储空间大约为邻接表所需存储空间的一半。不难看出：在顶点数不变的情况下，随着边数的增加，OBDD比邻接表节省存储空间越来越多。

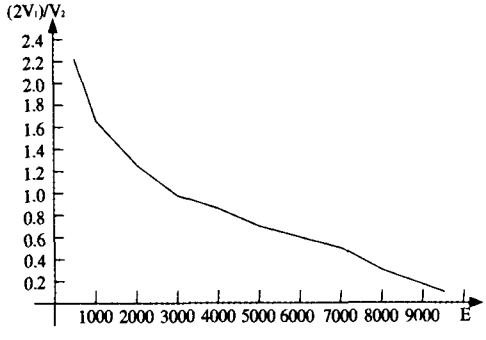

图3 OBDD与邻接表所占空间之比的变化曲线

结束语 本文讨论了有向图的有序二叉决策图存储结构，该存储结构的设计有别于传统的设计思想，它把有向图转化为布尔函数并用基于 OBDD 的图算法对有向图进行操作。基于 OBDD 存储结构的有向图的操作和搜索算法，适合于对大型有向图进行存储和操作。实验表明，该存储结构与传统的邻接表相比，在处理大规模有向图时，具有较高的存储效率。

参 考 文 献

1 周海岩. 图的关联链式存储结构. 计算机研究与发展，1997，34(10)：101～106

2 Akers S B. Binary Decision Diagrams. IEEE Transaction on Computer，1978，27(6)：509～516

3 Bryant R E. Graph-Based Algorithms for Boolean Function Manipulation. IEEE Trans. on Computers，1986，35(8)：677～691

4 Somenzi F. CUDD：CU decision diagram package release 2.3.1. http://vlsi.Colorado.edu / ～fabio /CUDD/cuddIntro.html，2001

（上接第 85 页）

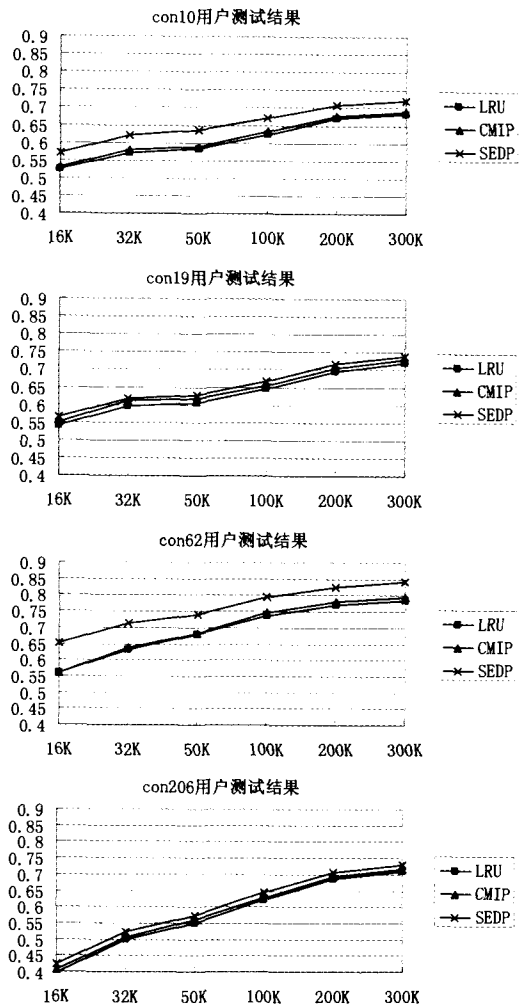

图 5 缓存命中率的测试结果

测试表明，SEDP 算法对于不同用户都能有很好的表现，其缓存命中率比 CMIP 算法提高 2%到 6%，比 LRU 算法提高 2%到 8%。

表 5 算法执行时间（单位：毫秒）

用户名	CMIP	SEDP
con1	44560.67	446.67
con6	313637.5	2136.5
con10	1091.17	807.83
con19	888.5	270.17
con62	11500.17	828
con206	751	272

表 6 算法占用内存空间（单位：字节）

用户名	CMIP	SEDP
con1	569000	38032
con6	93320	84524
con10	19112	77438
con19	12516	21194
con62	29684	65804
con206	5781	7388

运行上述算法所用时间如表 5 所示，从所用时间上看，SEDP 算法比 CMIP 算法的复杂度更低。运行上述算法所用内存大小如表 6 所示，数据挖掘所用的内存空间并不稳定。但是由于数据挖掘只在缓存命中率低的时候才运行，而且运行频率不高，所以短时间占用较多的内存空间对系统影响不大。

小结 本文主要研究无线局域网内的 Web 缓存技术，我们给出了适用于无线 LAN 的 Web 缓存系统 EasyCache 的实现，该缓存系统能支持手持设备的移动切换，并具有局域网内的缓存协作和负载平衡功能。我们在 EasyCache 中提出了一种基于数据挖掘的预取算法 SEDP，实验证明：SEDP 算法的缓存命中率比 CMIP 提高了 2%～6%，比 LRU 算法提高 2%～8%，而且 SEDP 算法的计算复杂度较低，适用于计算能力较低的移动手持设备。

参 考 文 献

1 Mallick M. Mobile and Wireless Design Essentials，Wiley，2003

2 Coulouris G，Dollimore J，Kindberg T. Distributed Systems Concepts and Design，Fourth edition，2005

3 Wessels D，Claffy K. ICP and the Squid Web Cache. IEEE Journal on Selected Areas in Communications，1998，16(3)

4 Fan Li，Cao Pei，Almeida J，Broder A Z. Summary Cache：A Scalable Wide-Area Web Cache Sharing Protocol. IEEE/ACM Transactions on Networking，2000，8(3)

5 Kuenning G，Popek G，Reiher P. An Analysis of Trace Data for Predicative File Caching in Mobile Computing. In：Proc. of the 1994 USENIX Conference，June 1994. 291～306

6 BU Web Trace. http://ita.ee.lbl.gov/html/contrib/BU-Web-Client.html

7 任鑫. 无线局域网的 Web 缓存研究与实现：[硕士论文]. 2006

8 Song Hui，Cao Guohong. Cache-Miss-Initiated Prefetch in Mobile Environments. Computer Communications，2005，28(7)：741～753