计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20160589
53(12):2783-2792,2016

大规模图数据的 k^2-MDD 表示方法与操作研究

董荣胜　张新凯　刘华东　古天龙
(广西可信软件重点实验室(桂林电子科技大学)　广西桂林　541004)
(ccrsdong@guet.edu.cn)

Representation and Operations Research of k^2-MDD in Large-Scale Graph Data

Dong Rongsheng, Zhang Xinkai, Liu Huadong, and Gu Tianlong
(*Guangxi Key Laboratory of Trusted Software* (*Guilin University of Electronic Technology*), *Guilin*, *Guangxi* 541004)

Abstract　Efficient and compact representation and operation of graph data which contains hundreds of millions of vertices and edges are the basis of analyzing and processing the large scale of graph data. Aiming at the problem, this paper proposes a representation of large-scale graph data based on the decision diagram, that is k^2-MDD, providing the initialization of k^2-MDD and the basic operation such as the edge query, inner(outer) neighbor query, finding out(in)-degree, adding(deleting) edge, etc. The representation method is optimized and improved on the basis of k^2 tree, and after dividing the adjacency matrix of graph into k^2, it is stored with the multi valued decision diagram, so as to achieve a more compact storage structure. According to the experimental results of a series of real Web graph and the social network graph data (cnr-2000, dewiki-2013, etc.) derived from the LAW laboratory at the University of Milan, it can be seen that the number of k^2-MDD' nodes is only 2.59%-4.51% of the k^2 tree, which achieving the desired effect. According to the experimental results of random graphs, it can be seen that the k^2-MDD structure is not only suitable for sparse graphs, but also for dense graphs. The graph data of k^2-MDD shows that both containing the compact and query efficiency representation of k^2 tree and realizing the efficient operation of the graph model can thus achieve the unity of description and computing power.

Key words　graph data; storage optimization; k^2-MDD; k^2 tree; decision diagram

摘　要　对包含亿万个顶点和边的图数据进行高效、紧凑的表示和操作是大规模图数据分析处理的基础.针对该问题提出了基于决策图的大规模图数据的一种表示方法——k^2-MDD,给出了 k^2-MDD 的构造过程以及图的边查询、外(内)邻查询、出(入)度查询、添加(删除)边等基本操作.该表示方法在 k^2 树的基础上进行优化与改进,对图的邻接矩阵进行 k^2 划分后,采用多值决策图进行存储,从而达到存储结构更为紧凑的目的.通过对来自米兰大学 LAW 实验室的一系列真实网页图和社交网络图数据的实验结果可以看出,k^2-MDD 结构在节点数上仅为 k^2 树的 2.59%～4.51%,达到了预期效果.通过对随机图的实验结果可以看出,k^2-MDD 结构不仅适用于稀疏图,同样也适用于稠密图.图数据的 k^2-MDD 表示,既具有 k^2 树表示的紧凑型和查询的高效性,又能实现符号决策图表示下图模式的高效操作,从而实现了描述和计算能力的统一.

收稿日期:2016-08-15;**修回日期**:2016-10-28
基金项目:国家自然科学基金项目(U1501252,61363070,61572146,61363030);广西高等学校高水平创新团队及卓越学者计划;桂林电子科技大学创新团队资助项目
This work was supported by the National Natural Science Foundation of China (U1501252, 61363070, 61572146, 61363030), the High Level Innovation Team of Guangxi Colleges and Universities and Outstanding Scholars Fund, and the Program for Innovative Research Team of Guilin University of Electronic Technology.
通信作者:古天龙(gu@guet.edu.cn)

关键词 图数据;存储优化;k^2-MDD;k^2 树;决策图

中图法分类号 TP311

随着移动互联网、物联网等技术的发展,众多新应用以前所未有的方式和速度产生并积累着大量数据.在众多类型的大数据中,图数据作为一种有效描述大数据的数据结构,扮演着越来越重要的角色[1-2].由于图数据的规模日益庞大,如何对图数据进行高效的存储以及如何对图数据进行高效的操作是当前面临的两大挑战.以社交网络为例,根据GlobalWebIndex统计,Facebook用户量已经超过11亿,平均每个人的好友超过100位,使用邻接表来存储所有用户的关系信息,需要接近1TB的存储空间[3];以互联网为例,根据中国互联网络信息中心(CNNIC)发布的《第37次中国互联网络发展状况统计报告》[4],截至2015年12月中国网页数量为2 123亿个,超链接数量据估计超过10^{13},使用邻接表来存储网页直接的链接关系信息需要超过16 TB的存储空间.同时随着用户量和信息量的快速增长,问题也只会变得越来越严峻.

为了对图数据进行紧凑表示,在传统的邻接矩阵表示法的基础上,Brisaboa等人[5]于2009年提出了基于k^2树(k^2-tree)的方法,树中的每一层对应于邻接矩阵或分块子矩阵的分块子矩阵,节点对应于邻接矩阵的分块子矩阵,生成的k^2树使用2个位向量$\boldsymbol{T}$和$\boldsymbol{L}$来存储,该方法不仅能够紧凑表示邻接矩阵,而且能实现邻接节点的正向或逆向高效查询操作[6].施佺等人[7-8]给出了k^2树表示方法的2种优化技术:启发式深度优先节点重排序和自适应修正k,使得所表示的结构更为紧凑,节点得到明显的减少.

但是,不论是k^2树还是施佺等人优化过的k^2树,在对大规模图数据表示时仍具有一定的局限性,具体表现在3个方面:

1) 当图的规模变大时,图内部本身就会存在大量的同构子图.同样地,当按照k^2树的思想把邻接矩阵进行划分后,也存在大量的相同的子矩阵.这就造成了k^2树内也存在大量的同构子树.

2) k^2树仅对稀疏图有效,当图变的稠密时,由于邻接矩阵内可被压缩的0节点变少,因此k^2树紧凑性也会变低.

3) k^2树未涉及动态图(需要添加或删除顶点、边以及子图等的图)的表示与操作[5].

因此,k^2树对上述图的结构特性尚缺乏必要的考虑,其在紧凑性上仍有较大的改善空间.针对k^2树目前存在的问题,有必要对其进行进一步的优化与改进,以得到一种更为紧凑并且适用面更广的图数据的表示方法.为此,本文提出一种基于决策图的大规模图数据的表示方法——k^2-MDD,该表示方法采用k^2树的思想对邻接矩阵进行划分,然后使用多值决策图进行存储,使k^2树中大量的同构子树所造成的冗余节点得到合并,从而达到存储结构更为紧凑的目的.该k^2-MDD的优点主要有4点:

1) 由于采用决策图结构来存储图数据,因此对于划分邻接矩阵时产生相同的子矩阵,即k^2树中的同构子树就自然地被合并了,最终生成的k^2-MDD结构将比k^2树紧凑得多.

2) k^2-MDD与k^2树所不同的是,它不论是0值还是1值,只要是同构的,都将被合并掉.因此,在表示稠密图时,k^2-MDD节点数反而会变少,结构变得更为紧凑.

3) 当使用决策图来存储图数据后,图的相关基本操作就可以转化为符号决策图的逻辑操作,这就为动态图数据的高效操作提供了可能性.另外这也使得基于k^2-MDD要比基于k^2树的图的查询操作更为简洁.

4) 由于k^2-MDD是基于决策图的结构,根据其本身结构的特性,该结构将更有利于子图查询、图同构、图/子图匹配以及多图匹配等方向的研究.

本文以有向无权图为例,对k^2-MDD的表示与操作进行讨论.本文提出的结构也可以拓展用于无向图以及加权图的存储与表示.下面给出k^2-MDD的表示形式、定义、性质、构造过程以及对于有向图的基本操作等.

1 有向图的k^2-MDD表示形式

首先给出k^2-MDD的定义:

定义1. k^2-MDD:图的邻接矩阵可以用一个将原始矩阵进行递归的k^2等分后构造的多值决策图(multi-valued decision diagram, MDD)[9]结构来表示,这样的MDD称之为k^2-MDD.

k^2-MDD描述了一个带有n个变量的离散多值函数,$f:D_1\times D_2\times\cdots\times D_i\times\cdots\times D_n\rightarrow S$,其中:

1) $n=\lfloor \log_k |V| \rfloor$, $|V|$指图中顶点个数. n 的值与对图的邻接矩阵递归划分的层数相同. k^2-MDD 的高度为 $n+1$.

2) D_i 代表递归到第 i 层时对(子)矩阵的划分. $D_i=\{1,2,\cdots,k^2\}$为所有变量的有限值域,与每次对(子)矩阵划分得到的 k^2 个子矩阵一一对应;S 为多值函数 f 的有限值域,即 k^2-MDD 终端节点的取值集合,其可能为布尔值(对应无权图)、有限整数集合或者有限实数集合(对应加权图).

3) k^2-MDD 的节点包括终端节点和非终端节点.

4) 非终端节点用 x_i 表示,包含 k^2 个指向其他节点的指针,这些指针和函数 f 对应,其形式化描述为

$$f_{x_i=c}=f(x_1,x_2,\cdots,x_{i-1},c,x_{i+1},\cdots,x_n). \quad (1)$$

k^2-MDD 的图形化描述如图 1 所示:

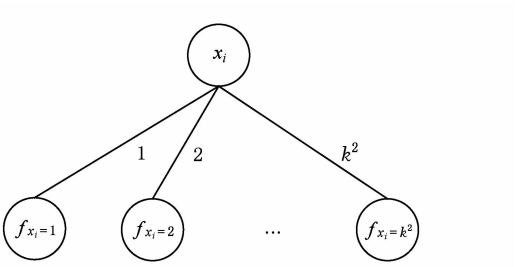


Fig. 1 Graphical description of k^2-MDD.

图 1 k^2-MDD 的图形化描述

从图 1 可以看出,给定多值变量 x_1 到 x_n 的一组取值,可以得到唯一的终端节点取值.

k^2-MDD 的化简规则与 MDD 相同,可以总结为 3 条:

规则 1. 合并相同终端节点. 同一属性的终端节点只保留一个,并删除其余相同属性的终端节点,原来指向这些已删除的终端节点的指针重定向到保留的终端节点上.

规则 2. 合并相同内部节点. 同一属性的内部节点(非终端节点)只保留一个,并删除其余相同属性的内部节点,原来指向这些已删除节点的指针重定向到保留的内部节点上.

规则 3. 删除冗余节点. 如果一个节点的所有指针都指向同一节点,那么该节点就是冗余节点,需将其删除,并将指向该节点的指针指向删除节点的孩子节点.

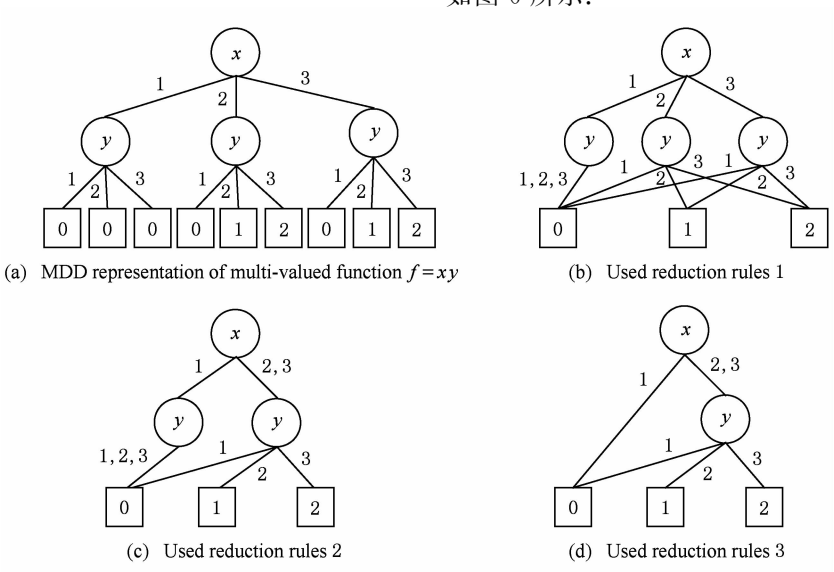
上述 3 条化简规则实例如图 2 所示,其中图 2(a)表示多值函数 $f=xy$,x 的取值范围为$x\in\{1,2,3\}$,y 的取值范围为 $y\in\{1,2,3\}$,函数 f 的值域为集合 $\{0,1,2\}$.

由 k^2-MDD 的定义可以看出,图的邻接矩阵中任一单元均对应于 n 个变量的唯一一组取值,根据这组取值可以得到唯一函数值,即终端节点的值,并且该值与原始矩阵中对应单元格的元素值相等.

本文以无权图进行说明,因此函数 f 的值域取布尔值. 对图中顶点数量等于 k 的若干次幂和图中顶点数量不等于 k 的若干次幂 2 种典型情况分别进行举例说明,其中 $k=2$.

例 1. 顶点数量等于 k 的若干次幂.

对于图 3 所示的有向图,其邻接矩阵如图 4 所示,k^2 树如图 5 所示. 显然,图 5 中存在同构子树,如图 6 所示.



(a) MDD representation of multi-valued function $f=xy$ (b) Used reduction rules 1

(c) Used reduction rules 2 (d) Used reduction rules 3

Fig. 2 Reduction rules.

图 2 化简规则

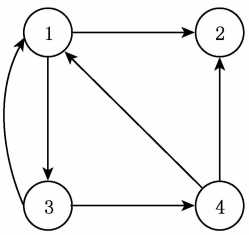


Fig. 3 Graph structure.

图 3 图结构

0	1	1	0
0	0	0	0
1	0	0	1
1	1	0	0

Fig. 4 Adjacency matrix of example 1.

图 4 例 1 的邻接矩阵

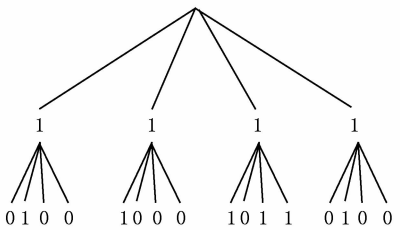


Fig. 5 k^2 tree of example 1.

图 5 例 1 的 k^2 树

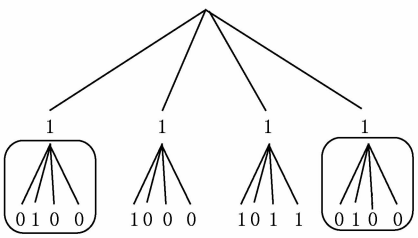


Fig. 6 Isomorphic subtrees in k^2 tree of example 1.

图 6 例 1 的 k^2 树中的同构子树

例 2. 顶点数量不等于 k 的若干次幂.

当图中顶点数量不等于 k 的若干次幂时,可以通过增加邻接矩阵的行数和列数(新增的元素全都标记为 0)来满足条件[3]. 如图 7 中的邻接矩阵,其原始图中有 11 个顶点,原始矩阵是 11 行乘 11 列的矩阵,通过增加 5 行 5 列 0 数据来使得矩阵可以被 k^2 等分. 从 k^2 树原理及图 8 可以知道,这样增加全部为 0 的行列对最终生成的 k^2 树产生的影响非常小. 显然,图 8 的 k^2 树也存在同构子树,图 9 标出了其中一部分.

0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 7 Adjacency matrix of example 2.

图 7 例 2 的邻接矩阵

当图的规模变大时,k^2 树中相同的子树会变得越来越多,另外还有大量的 0 节点. 这种现象在小规模图中也有体现,比如上述 2 个例子.

为了减少 k^2 树中的节点数量,可以将 k^2 树中的同构子树进行合并. 而这正好是决策图[10]的优势. 另外,这种合并的过程与 MDD 化简过程恰好吻

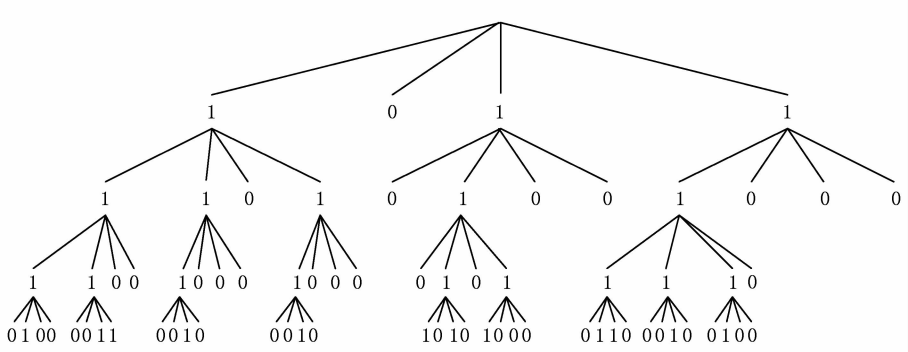


Fig. 8 k^2 tree of example 2.

图 8 例 2 的 k^2 树

合.对于无权图,使用布尔型 MDD(终点要么是真要么是假);对于加权图,使用整数或者实数型 MDD(终点是整数或者实数).本文使用无权图进行说明,MDD 中所有变量的有限值域均为$\{1,2,\cdots,k^2\}$.

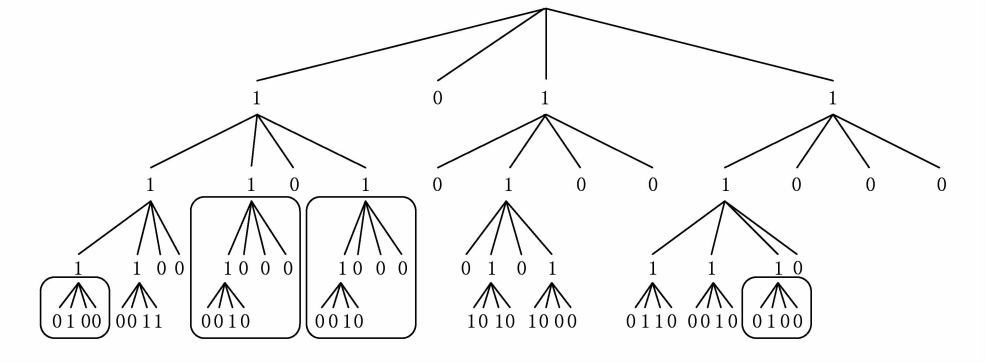


Fig. 9 Isomorphic subtrees in k^2 tree of example 2.

图 9 例 2 的 k^2 树中的同构子树

将上述 2 个例子的 k^2 树转换为 MDD 后分别如图 10 和图 11 所示.图 10、图 11 中节点内的数字仅代表该节点的索引号,并没有其他意义,图 10、图 11 中左侧的数字为图中节点的层次序号.

从以上 2 例可以看出,从 k^2 树转换为 MDD 后同构子树都被合并了,节点数量大大减少.例 1 从 21 个节点减少为 6 个节点,例 2 从 73 个节点减少为 16 个节点.

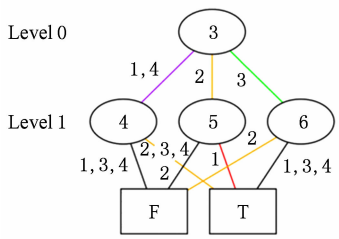


Fig. 10 The MDD of example 1.

图 10 例 1 的 MDD

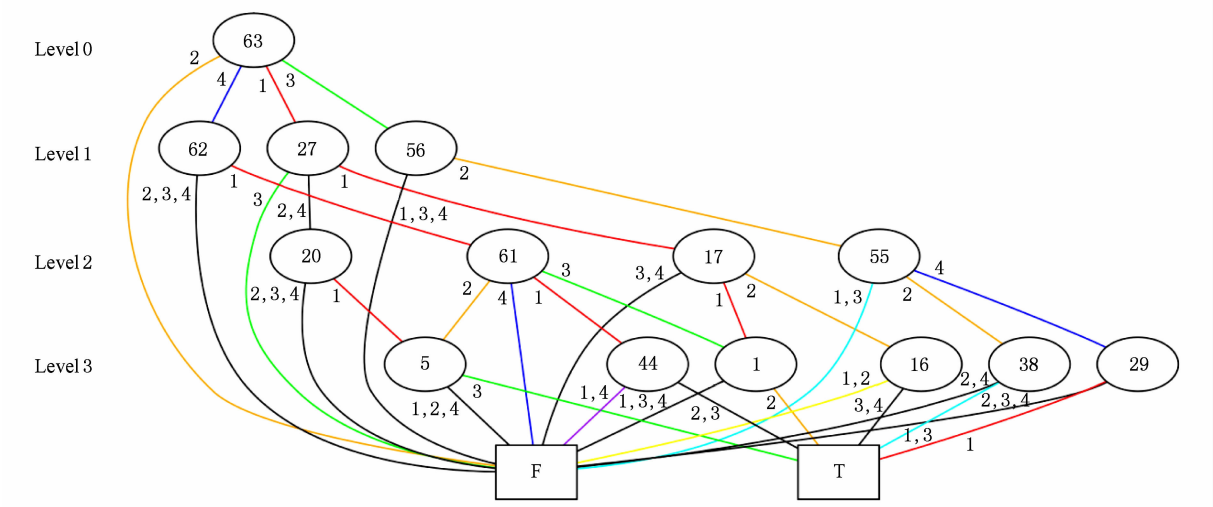


Fig. 11 The MDD of example 2.

图 11 例 2 的 MDD

2 构造 k^2-MDD 的算法过程

构造 k^2-MDD 的过程分为对图的顶点进行编码、对边进行编码、根据边的编码来构造 k^2-MDD 3 个步骤,具体如下:

1) 对图的顶点编码

对于有向图 $G=(V,E)$,可以对顶点进行 n 位二进制编码,$n=\lfloor \log_k |V| \rfloor$.对于编号为 N 的顶点,将顶点总数$|V|$进行递归 k 等分方式进行编码.因此当 $k=2$ 时,在顶点的 n 位编码中每一位都是

2 种状态之一(0 或 1).顶点编码过程的伪代码如算法 1 所示.根据该编码规则对图 3 所示图的顶点编码后如图 12 所示:

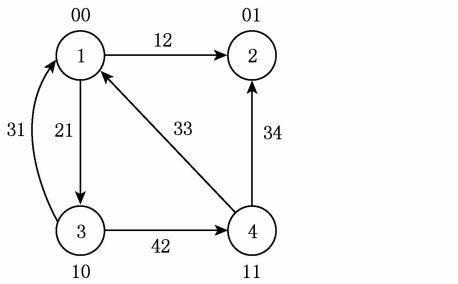


Fig. 12 Coding vertexs and coding edges.

图 12 顶点编码和边编码

算法 1. 对图的顶点进行编码.

输入:要进行编码的顶点的编号 *nodeID*;

输出:对应该顶点的编码数组 *nodeCode*[].

① *EncodingNode*(*nodeCode*[],*nodeID*)
/*将编号为 *nodeID* 的顶点编码并存到数组 *nodeCode*[]中*/

② *codeNum*←*CeilingLog_2*(*nodeNum*);
/*codeNum* 为编码长度,*nodeNum* 为顶点数量*/

③ *low*←1;

④ *high*←*Pow*(2,*codeNum*); /*取不小于 *nodeNum* 的 2 的若干次幂*/

⑤ *i*←1;

⑥ While (*low*≤*high*) Do /*二分编码*/

⑦ *mid*←(*low*+*high*)/2;

⑧ If *nodeID*≤*mid* Then

⑨ *nodeCode*[*i*]←0;

⑩ *high*←*mid*−1;

⑪ Else

⑫ *nodeCode*[*i*]←1;

⑬ *low*←*mid*+1;

⑭ EndIf

⑮ *i*←*i*+1;

⑯ EndWhile

2) 对图的边编码

有向边可理解为顶点之间的关系,顶点 v_0 到顶点 v_1 的边可用特征函数 $E(v_0,v_1)$来描述.当 $k=2$ 时,设 $\boldsymbol{X}=(x_1,x_2,\cdots,x_n)$,$\boldsymbol{Y}=(y_1,y_2,\cdots,y_n)$是图中顶点的编码向量,则顶点 X 到顶点 Y 边的特征函数表示为

$$E(\boldsymbol{X},\boldsymbol{Y}):\{0,1\}^n\times\{0,1\}^n\rightarrow\{1,2,3,4\}^n,$$

即每一位上 2 个顶点的 2 种状态会组合出 4 种状态.因此,边的编码长度依然是 n 位,每一位是 4 种状态之一(1,2,3 或 4).边编码过程的伪代码如算法 2 所示.根据该编码规则对图 3 所示图的边编码后如图 12 所示.

算法 2. 对图的边进行编码.

输入:要进行编码的边的起止顶点编号 *nodeFrom* 和 *nodeTo*;

输出:对应该条边的编码数组 *edgeCode*[].

① *EncodingEdge*(*edgeCode*[],*nodeFrom*,*nodeTo*) /*根据边的起止顶点得到边的编码*/

② *EncodingNode*(*nodeCodeA*[],*nodeFrom*)
/*对边的起始顶点进行编码*/

③ *EncodingNode*(*nodeCodeB*[],*nodeTo*)
/*对边的终止顶点进行编码*/

④ For *i*=1 To *codeNum* Do /*根据起止顶点的编码来对这条边进行编码*/

⑤ *edgeCode*[*i*]←*nodeCodeA*[*i*]×2+*nodeCodeB*[*i*]+1;

⑥ EndFor

3) 根据边编码的集合来构造 k^2-MDD

根据 k^2-MDD 的定义,我们可以构造一个含有 n 个变量的 k^2-MDD,然后令这 n 个变量的值等于边编码集合里的值时,函数值为 T,否则为 F.这样就得到了与有向图 G 对应的 k^2-MDD.例如,根据图 12 中的边编码,可以得到图的 k^2-MDD 结构如图 10 所示.在构造 k^2-MDD 时可以借助 MEDDLY (multi-terminal and edge-valued decision diagram library) 函数库[11].MEDDLY 函数库是为操控 MDD 提供的一个 C/C++开源项目,由爱荷华州立大学在 Linux 平台下开发,其中提供了丰富的 MDD 构造以及操作的函数.例如:可以使用 *createVariablesBottomUp*()函数来创建将要构造 MDD 的变量个数以及每个变量的取值范围;使用 *createEdge*()函数来根据给定的一组或多组变量的值来生成一个 MDD;使用 *apply*()函数以及 UNION 运算符来将 2 个 MDD 进行合并.通过 MEDDLY 函数库,根据边编码我们可以很方便地生成有向图对应的 k^2-MDD.构造 k^2-MDD 的伪代码如算法 3 所示.

算法 3. 构造 k^2-MDD.

输入:图的所有边;

输出:与输入图对应的 k^2-MDD.

① *CreateK2MDD*() /*根据边生成 k^2-MDD*/

② For *i*=1 To *codeNum* Do /*设置变量

```
        个数以及每个变量的值域 */
③       Bounds[i]←4;
④   EndFor
⑤   createVariablesBottomUp (bounds [], codeNum); /* 创建变量 */
⑥   EncodingEdge(edgeCode[], nodeFrom[1], nodeTo[1]); /* 对第 1 条边进行编码 */
⑦   createEdge(edgeCode[],1,all); /* 根据第 1 条边的编码生成一个初始化 MDD */
⑧   For i=2 To edgeNum Do
⑨       EncodingEdge(edgeCode[], nodeFrom[i], nodeTo[i]); /* 对第 i 条边进行编码 */
⑩       createEdge(edgeCode[],1,rest); /* 根据第 i 条边编码生成 MDD 并存到 rest */
⑪       apply(UNION,all,rest,all); /* 把新生成的 rest 合并到已有的 all 里 */
⑫   EndFor
⑬   K2 MDD←all. /* 将所有边生成的 MDD 都合并后就得到最终的 k²-MDD */
```

3 基于 k^2-MDD 的有向图的操作

下面给出在 k^2-MDD 存储结构下有向图的一些基本操作.

1) 边查询

根据边的起止顶点 v_1 和顶点 v_2 的编号得到该条边的编号 $E(v_1,v_2)$,在图的 k^2-MDD 中检测 $E(v_1,v_2)$的函数值.若值为 T,则该边存在,否则不存在.MEDDLY 库中提供的 *INTERSECTION* 运算符可以帮我们完成此操作. *INTERSECTION* 用来求 2个 MDD 的交运算.因此,我们只要将原图的 k^2-MDD 与要查询的边生成的 k^2-MDD 进行 *INTERSECTION* 运算,查看运算结果是否为空即可.边查询的伪代码如算法 4 所示.

算法 4. 边查询算法.

输入:要查询的边的起止顶点编号 *nodeFrom* 和 *nodeTo*;

输出:该边是否存在.

```
① edgeQuery(nodeFrom, nodeTo)   /* 根据边的起止顶点查询图中是否存在该边 */
②   EncodingEdge(edgeCode[], nodeFrom, nodeTo); /* 对边进行编码 */
③   createEdge(edgeCode[],1,tmp); /* 生成这条边的 MDD */
④   apply(INTERSECTION, K2MDD, tmp, res); /* 进行交运算,结果存到 res 中 */
⑤   If res.getNode()=0 Then   /* 如果结果为空,则边不存在;否则存在 */
⑥       Print(不存在);
⑦   Else
⑧       Print(存在);
⑨   EndIf
```

2) 外邻查询(包括求出度)

借助边查询,将要进行外邻查询的顶点赋值为 v_1,图中所有顶点依次赋值为 v_2,检测 $E(v_1,v_2)$的函数值.若值为 T,则当前 v_2 是一个外邻点,否则不是.通过统计外邻点个数可以得到该顶点的出度.

3) 内邻查询(包括求入度)

查询方法与外邻查询类似,只不过要查询的点赋值为 v_2,图中所有顶点依次赋值为 v_1.

4) 增加边

根据要增加边的起止顶点 v_1 和 v_2 的编号得到该条边的编号 $E(v_1,v_2)$,生成该条边的 k^2-MDD;然后与原图的 k^2-MDD 进行 UNION 运算,即得到增加了该边的新图的 k^2-MDD.

5) 删除边

根据要删除边的起止顶点 v_1 和顶点 v_2 的编号得到该条边的编号 $E(v_1,v_2)$,生成该条边的 k^2-MDD;然后将原图的 k^2-MDD 与要删除边的 k^2-MDD 进行 DIFFERENCE 运算,即得到删除了该边的新图的 k^2-MDD. DIFFERENCE 是求 2 个 MDD 的差运算,执行函数 *apply*(DIFFERENCE,A,B,*res*)后,$res=\{x|x\in A$ 且 $x\notin B\}$.

根据上述图的基本操作,可以很方便地拓展到图的一些复杂操作,比如图中宽度优先搜索、求最短路、网络流等.

4 算法复杂度分析

由边编码集合构造 k^2-MDD 以及基于 k^2-MDD 的有向图的添加和删除边,由于涉及到 MDD 的初始化、生成、求交集、求并集及化简等运算过程,在文献[9]中已经进行了复杂度分析,这里不再分析.

在构造 k^2-MDD 时,对图的单个顶点编码的时间复杂度为 $O(\log_k|V|)$;同理,对图的单条边编码的时间复杂度也是 $O(\log_k|V|)$;因此对图的所有边

进行编码的时间复杂度即为 $O(|E|\log_k|V|)$.

在基于 k^2-MDD 的有向图的操作中,由于 k^2-MDD 的高度为 $n=\lfloor\log_k|V|\rfloor$,因此边查询的时间复杂度为 $O(\log_k|V|)$;由此可得外邻查询(包括求出度)和内邻查询(包括求入度)的时间复杂度为 $O(|V|\log_k|V|)$.

对于大规模图数据存储于操作,在实际应用中,通常评价指标是存储结果和查询时间,对于构造时的复杂度一般并不关心.

5 实验与分析

本文利用 MEDDLY 库,采用 C++语言实现所提出的算法.实验用机器配置的软件平台为 Ubuntu14.04 LTS 64 位操作系统(内核 Linux 3.19.0-61-generic),硬件平台为 Intel® Core™ i5-4690 CPU @ 3.50 GHz 处理器,8 GB 内存.测试数据采用公开的真实数据集和随机数据集.真实数据集使得实验结果具有更好的代表性,随机数据集可以看出存储结构在不同规模图数据下的节点数变化趋势.

5.1 基于真实数据集的实验

表 1 列出了测试中使用的真实网页图和社交网络数据,数据均来自米兰大学 LAW(Laboratory for Web Algorithmics)[12].表 1 分别给出了这些数据集的顶点数、边数、顶点数与边数的比值以及这些数据集在网站的文件名.

Table 1 Description of Testing Real Graphs

表 1 测试使用的真实数据集

Graphs	Data Sets	Number of Nodes N	Number of Edges E	E/N	File Name
Web Graphs	cnr	325 557	3 216 152	9.88	cnr-2000
	in	1 382 908	16 917 053	12.23	in-2004
	uk	4 769 354	50 829 923	10.66	uk-2014-host
	indochina	7 414 866	194 103 311	26.18	indochina-2004
Social Network Graphs	enron	69 244	276 143	3.99	enron
	dblp-1	326 186	1 615 400	4.95	dblp-2010
	dblp-2	986 324	6 707 236	6.80	dblp-2011
	dewiki	1 532 354	36 722 696	23.96	dewiki-2013

在网页图中同一个域内的网页之间的链接数比较多,而域之间的链接数就比较少,其具备 2 个特征:

1) 本地性.大多数链接都是域内的,它们通常会指向字典序比较靠近的页面.

2) 相似性.在字典序中邻近页面的邻居集合也是相似的.

而社交网络图并不可以直接对其中的节点进行排序,但是也包含着类似于网页图体现的特征,例如同一个社区内的好友关系数量比较多,而社区之间的好友关系数量就比较少.

对于这些真实数据集的特性,本文提出的 k^2-MDD 结构正好可以发挥其同构子树合并的优势.

针对这些数据集,分别使用 k^2 树、k^2-MDD 以及有序二叉决策图(ordered binary decision diagram, OBDD)[10]共3 种方法进行了测试.表 2 给出了这些数据集在上述 3 种方法下的实验结果,表 2 中的数据为这 3 种存储结构的节点数.

Table 2 The Experimental Results

表 2 实验结果

Graphs	Data Sets	k^2 Tree	OBDD	k^2-MDD
Web Graphs	cnr	11 246 165	2 435 522	342 893
	in	49 442 029	10 652 901	1 205 089
	uk	464 588 901	59 830 211	14 793 248
	indochina	467 001 541	63 727 936	7 974 153
Social Network Graphs	enron	2 490 345	242 821	125 481
	dblp-1	12 018 925	3 260 150	621 021
	dblp-2	69 557 893	12 269 732	3 310 652
	dewiki	615 969 633	50 403 738	18 895 096

5.2 基于随机数据集的实验

除上述真实数据集外,我们随机生成了 11 组图数据,顶点数均为 1 000,边数分别为 10 000,100 000,200 000,…,900 000,1 000 000. 这些图中顶点之间的连接关系完全随机,因此没有网页图和社交网络的特性. 需要说明的是,当边数为 1 000 000 时,图为完全图. 图 13 给出了 k^2 树和 k^2-MDD 在这些随机数据集上的实验结果.

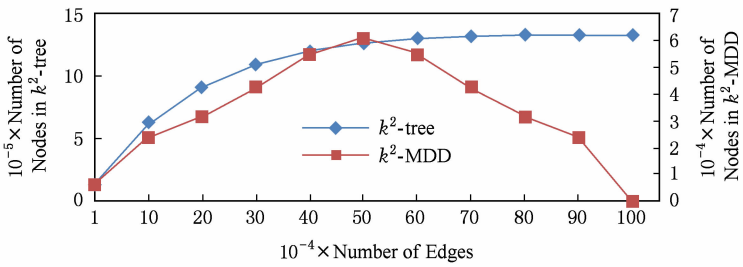


Fig. 13 The experimental results of the random graphs.

图 13 随机图实验结果

5.3 实验分析

对于真实数据集,从表 2 的实验结果中的数据可得,在选用的 4 组网页图上,k^2-MDD 的节点数平均仅为 k^2 树的 2.59%;在选用的 4 组社交网络上,k^2-MDD 的节点数平均仅为 k^2 树的 4.51%. 可以看出使用 k^2-MDD 结构来存储图可以大大减少节点数量,达到了预期效果. 另外,还可以看到,OBDD 结构在存储图时,节点数也比 k^2 树大大减少,这也体现出了决策图在节点合并方面的优势.

对于随机数据集,从图 13 的实验结果中的数据可以看出,在图的顶点数不变的情况下,随着边数增多,即图变的越来越稠密,k^2 树的节点数在持续增加,而 k^2-MDD 的节点数在图的边数达到完全图的一半之后开始减少. 这说明 k^2 树只适用于稀疏图,而 k^2-MDD 不仅适用于稀疏图还适用于稠密图. 因为 k^2 树只对某节点的指针全部指向 0 节点的情况进行了合并,所以造成了这样的结果.

在查询时间方面,由于边查询的时间复杂度与 k^2 树相同,而同等规模的图,k^2-MDD 与 k^2 树结构的高度也相同,因此查询时间不相上下. 通过对真实数据集和随机数据集进行实验验证也确是如此. 因此本文未给出 k^2-MDD 与 k^2 树在查询时间方面的数据对比.

6 结束语

本文提出了一种新的图数据的表示方法——k^2-MDD,这种存储结构的设计跟传统思想有所区别. k^2-MDD 把有向图的邻接矩阵进行 k^2 划分后,转化为多值布尔函数并使用基于 MDD 的相关逻辑运算对有向图进行操作. 实验表明:在处理真实数据集时,k^2-MDD 的节点数比 k^2 树大大减少,优势相当明显;在处理随机数据集时,k^2-MDD 不仅适用于稀疏图,同样也适用于稠密图. 当存储结构变得紧凑之后,对于图的相关计算的复杂度也会相应地变低. 文中给出了 k^2-MDD 对图的边进行增加与删除的操作,稍加拓展,即可得到对顶点的添加与删除以及对子图的添加与删除,因此本结构适用于动态图. 图数据的 k^2-MDD 表示,既具有 k^2 树表示的紧凑型和查询的高效性,又能实现符号决策图表示下图模式的高效操作,从而实现了描述和计算能力的统一.

参 考 文 献

[1] Angles R, Gutierrez C. Survey of graph database models [J]. ACM Computing Surveys, 2008, 40(1): 1-39

[2] Robinson I, Webber J, Elfrem E. Graph Databases [M]. Sebastopol, CA: O'Reilly Media Press, 2015

[3] Zhang Yu, Liu Yanbing, Xiong Gang, et al. Survey on succinct representation of graph data [J]. Journal of Software, 2014, 25(9): 1937-1952 (in Chinese)
(张宇, 刘燕兵, 熊刚, 等. 图数据表示与压缩技术综述[J]. 软件学报, 2014, 25(9): 1937-1952)

[4] China Internet Network Information Center. The 37th statistical report on Internet development in China [EB/OL]. 2016 [2016-01-31]. http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/201601/t20160122_53271.htm (in Chinese)
(中国互联网络信息中心. 第 37 次中国互联网络发展状况统计报告[EB/OL]. 2016[2016-01-31]. http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/201601/t20160122_53271.htm)

[5] Brisaboa N R, Ladra S, Navarro G. k^2-trees for compact Web graph representation [C] //Proc of the 16th Int Symp on String Processing and Information Retrieval. Berlin: Springer, 2009: 18-30

[6] Brisaboa N R, Ladra S, Navarro G. Compact representation of Web graphs with extended functionality [J]. Information Systems, 2014, 39(1): 152-174

[7] Shi Quan, Xiao Yanghua, Lu Yiqi, et al. Towards optimizing k^2-tree for large-scale graph storage [J]. Application Research of Computers, 2011, 28(7): 2488-2491 (in Chinese)
(施佺, 肖仰华, 鲁轶奇, 等. 基于 k^2 树的大图存储优化研究[J]. 计算机应用研究, 2011, 28(7): 2488-2491)

[8] Shi Quan, Xiao Yanghua, Bessis N, et al. Optimizing k^2-trees: A case for validating the maturity of network of practices [J]. Computers & Mathematics with Applications, 2012, 63(2): 427-436

[9] Srinivasan A, Ham T, Malik S, et al. Algorithms for discrete function manipulation [C] //Proc of ICCAD-90. Piscataway, NJ: IEEE, 1990: 92-95

[10] Gu Tianlong, Xu Zhoubo. Ordered Binary Decision Diagram and Its Application [M]. Beijing: Science Press, 2009 (in Chinese)
(古天龙, 徐周波. 有序二叉决策图及其应用[M]. 北京: 科学出版社, 2009)

[11] Iowa State University Research Foundation. MEDDLY: Multi-terminal and edge-valued decision diagram library, Version 0.11.486 [CP/OL]. 2014 [2015-10-02]. http://meddly.sourceforge.net/index.html

[12] Department of Computer Science, University of Milan. Laboratory for Web algorithmics [DB/OL]. 2015 [2015-11-01]. http://law.di.unimi.it/datasets.php

Dong Rongsheng, born in 1965. Professor. Senior member of China Computer Federation. His main research interests include protocol engineering and wirless sensor networks.

Zhang Xinkai, born in 1991. Master candidate. His main research interests include graph data representation and symbolic technology.

Liu Huadong, born in 1978. Lecturer. His main research interests include graph data representation and symbolic technology.

Gu Tianlong, born in 1964. PhD, professor, PhD supervisor. Senior member of China Computer Federation. His main research interests include formal method, formal computing and knowledge engineering.