基于 BDD的图表示及其算法*

吕关锋，苏开乐，林 瀚，骆翔宇，陈清亮，岳伟亚
（中山大学 计算机科学系，广东 广州 510275）

摘 要：给出基于二元判决图 BDD的无权图和有权图的符号化表示，同时给出该表示下的算法设计及实现，并以连通度算法和最短路径算法作为例子。

关键词：BDD；符号化算法；连通度；最短路径
中图分类号：TP301.6 文献标识码：A 文章编号：0529-6579（2006）01-0020-05

二元判决图（reduced ordered binary decision diagram）BDD[1]是表示一个基于有序变量集上的布尔函数，由于其规范性和紧凑性，使得空间需要大大降低，从而提高操作速度。本文的主要工作是解决如何使用 BDD来表示图，尤其是有权图，并在此基础上设计算法，以达到利用 BDD的优势来设计图论算法的问题，另外将给出二个基于 BDD的符号化图论算法例子：无权图的连通度算法和有权图的最短路径算法，并与它们的经典算法作比较：融合节点法[2]和 Dijkstra算法[3]。

本文第 1节给出 BDD的基本概念；第 2节为无权图的 BDD表示，连通度算法，以及实验和比较；第 3节给出有权图的 BDD表示和最短路径算法，以及实验和比较。最后为相关工作介绍及结束语。

1 二元判决图（BDD）及其应用

一个二元判决图 BDD[1]是一个有向无圈图，表示一个基于一有序变量集上的布尔函数。BDD是一种规范且紧凑的布尔函数表达。例如布尔公式 a+b的 BDD表示如图 1所示。

另外，对于布尔操作，BDD有较高的效率，有兴趣的读者可参考文献[1]。

最近十几年来，在符号化模型检测领域，基于 BDD表示的高效查找技术已经发展起来[4-9]，使得模型检测工具能够验证状态数目非常大的系统[5]。

最近，BDD被应用到知识表达和推理领域[10,11]，知识的模型检测及安全协议的验证领域[12]，都取得了非常好的结果。

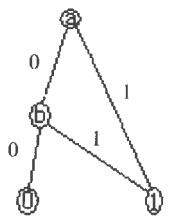

图 1 逻辑公式 a+b的 BDD
Fig. 1 BDD of logic formula a+b

2 无权图的 BDD表示

2.1 图的基本定义

为方便下面叙述，以下简要给出图的一些基本定义。

定义 1[2] 称 $G=(V, E)$是一个（有向）图，如果

（1）V是一个非空有限集合；

（2）E是 V中元素的无序（有序）对所组成的有限集合。

并把 V的元素叫做（有向）图的顶点，E的元素叫做图的（有向）边。用 $E(G)$、$V(G)$分别表示图 G的所有边和节点集合。设 H，G为图，且 $V(H)\subseteq V(G)$，$E(H)\subseteq E(G)$，则称 H为 G的子图。设 $v\in V(G)$，则与 v关联的边的数目称为 v的度。

* 收稿日期：2005-01-03
基金项目：国家自然科学基金资助项目（60496327，10410638，60473004）；德国研究基金资助项目（446 CHV 113/240/0-1）；广东省自然科学基金资助项目（04205407）
作者简介：吕关锋（1973年生），男，博士，现为北京工业大学计算机系讲师；E-mail：lvgf@yahoo.com

定义 2[2]　设 G是一个图，如果 G的一个节点和边的非空有限交错序列 $W=v_0e_1v_1\cdots e_kv_k$，K为自然数，满足 $e_i=(v_{i-1}, v_i)$，$i=1, 2, \cdots, k$，则称 W为一条 v_0-v_k通道，v_0为起点，v_k为终点。

定义 3[2]　$G=(V, E)$是一个图，图中二节点 u，v称为连通的，如果在 G中有 $u-v$通道。称图 G是连通的，如果 G的任二节点都是连通的。图 G的节点间的连通关系是一个等价关系，V可按连通关系分解为等价类(记为 k个)，即图 G可分解为 k个不相交子图的并，每个子图称为连通支，不相交子图数 k称为连通支数。

2.2　无权图的 BDD表示

设图 G有 n个节点，则可以定义 k个布尔变量 $x_1, x_2, \cdots, x_k$，其中 k为不小于 $\log_2(n)$的自然数。使用变量集 $X: x_1, x_2, \cdots, x_k$对每个节点编码，记为 $E(u)$。为了表示节点间的关联关系，引入另一组变量 $X': x_1', x_2', \cdots, x_k'$。把 $E(u)$称为 $E(u)$的后继编码，这里 $E'(u)$是把 $E(u)$中 $x_1, x_2, \cdots, x_k$用相应的 $x_1', x_2', \cdots, x_k'$替换，例如 x_1用 x_1'替换。如果节点 i和 j之间存在边 $E(i, j)$，则生成 BDD b_{ij}:

$$b_{ij}=E(i)\wedge E'(j)$$

图 G的所有关联关系可表示如下：

$T(G)=\vee b_{ij}$，其中 i，j取遍 1到 n。如果 i和 j不相邻，则 $b_{ij}=$false。

此时 BDD $T(G)$已经能够把无权图 G表示出来。

2.3　连通度的融合节点算法

对于一个图，求其连通度是一个常常遇到的问题。下面介绍判断无向图的连通度的一个经典的算法，即融合节点法[2]。

定义 4[2]　设 $(u, v)\in E(G)$，若用一个新节点 w代替节点 u与 v，而与节点 u、v关联的边都与新节点 w关联，则融合节点 u与 v，并称 w为融合点。

由定义，一次融合节点数目减少 1。融合节点法的基本步骤是任选图 G的一个节点 v，融合与节点 v相邻的所有节点得融合点，仍记为 v，再融合与其相邻的节点，重复这一过程，直到 G没有节点能融合到 v为止。此时，若 G的所有顶点都被融合，则 G是连通的。否则选择 G的没被融合的节点，重新开始融合过程。当融合不能进行时终止。此时，有多少个融合点就有多少个连通支。每个融合点所包含的 G的节点集的导出子图就是 G的连通支。融合节点法的时间复杂度为 $O(n^2)$。

2.4　基于 BDD的连通度符号化算法

算法的思想是：任选图 G的节点 u，计算与其关联的节点集 $S(u)$，再从 $S(u)$出发计算与 $S(u)$关联的节点集(即对于该集中的任一节点，$S(u)$中都存在节点关联它)，重复此过程，直到 $S(u)$不变为止。如果 $S(u)=V(G)$，则该图是连通的。否则选择 G的没被处理过的节点，重新开始上面计算过程。直到计算不能进行时终止，此时找到 G的所有的连通支。此过程跟融合节点法类似，但最重要的区别是：

(1) 在整个计算过程中，符号化算法是基于 BDD表示(图的隐式表示)，而融合顶点法是基于图的显式表示(如邻接矩阵)。

(2) 符号化算法的运算基本单位是节点的集合，而融合节点法是一个一个节点地处理。

计算节点集 S关联的节点集通过下式得到：

$$\mathrm{Img}(S)=(\mathrm{Exist}(X)(T(G)\wedge S))[X'X]$$

其中，$(\mathrm{Exist}(X)(T(G)\wedge S))$表示对 $(T(G)\wedge S))$的变量集 X求逻辑存在，$[X'X]$表示用 X'中的变量替换运算结果中的 X中的变量。算法如下：

(1) 读入图(如邻接矩阵)，生成表示图的 BDD，即 T。令 $k=1$，即选择节点 1。

(2) 对 k做记录，令 b为表示节点 k的 BDD。

(3) 计算 $\mathrm{Img}(b)$，赋给 t，计算 $\mathrm{Img}(t)$，重复，直到 $\mathrm{Img}(t)$不变。

(4) 对于每个节点 u，设其 BDD编码为 $E(u)$，如何 $E(u)$逻辑蕴涵 $\mathrm{Img}(t)$，则对 u做记录。

(5) 在未被记录的选一节点 k，转 2，如果都被记录，则转 6。

(6) 输出连通支，算法结束。

备注：在步骤 1中生成 BDD，即 T，是需要时间的，它实际上已经把图的所有关联关系表示出来，即为图的另一种表示方式，就像邻接矩阵一样。所以对于一个图，表示其的 BDD只需生成一次，当然也可以保存在外存(如硬盘)上。所以在对图进行具体算法时，无需考虑生成 T的时间，就像使用邻接矩阵计算时，不会把生成邻接矩阵的时间也考虑在内。

2.5　连通度算法实现与比较

我们用 C++编程实现了二算法，对大量随机生成的图进行了运算，实验所用的机器是主频为 2.4 GHz、内存为 512 M的 PC机。当图的平均度数不是特别大时，符号化算法明显比融合节点法(用邻接矩阵表示)效率高。特别是当图越大时，效率反差越大。3个具有代表性的实验结果如图 2、3和 4。通过对大量实验结果的观察及分析得出：

（1）图的平均度数大小对于融合节点算法影响很小，这是由其算法本身决定的。

（2）当图的平均度数比较小时，符号化算法的效率越好。实验表明当图的平均度数为 5到几十之间时（稀疏图），效率最好，在此情况下当图节点数达到 12 000时，符号化算法已经比融合顶点算法快一个数量级，而且这种优势将随着图节点数增加进一步地扩大，参见图 3。

（3）当图平均度数增大时，符号化算法的优势将缩小。这是因为此时图的 BDD 表示变大的原因。

（4）当图的节点数达到一定的值时，图的隐式表示 BDD 在内存中仍然能够表示，但图的显式表示（如邻接矩阵）将超过计算机的内存，故融合节点算法在计算过程中不得不与外存交互，而外存速度远远慢于内存速度，此时符号化算法的优势就会更加明显。这主要得益于 BDD的紧凑性。

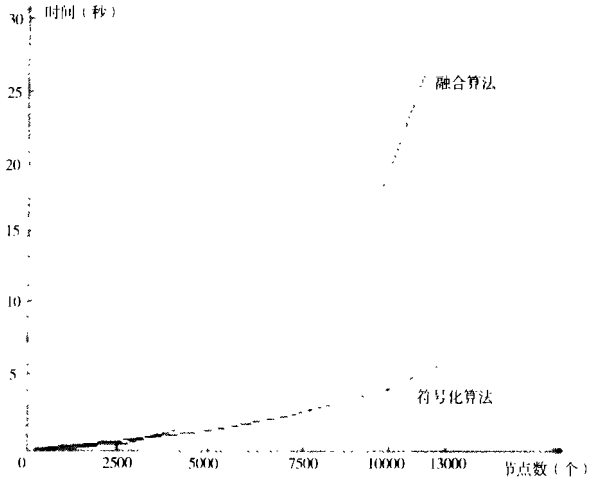

图 2 图平均度数为 3

Fig. 2 Average degree is 3

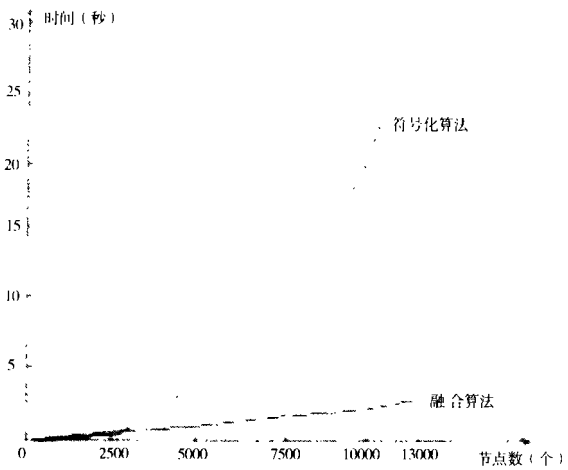

图 3 图平均度数为 10

Fig. 3 Average degree is 10

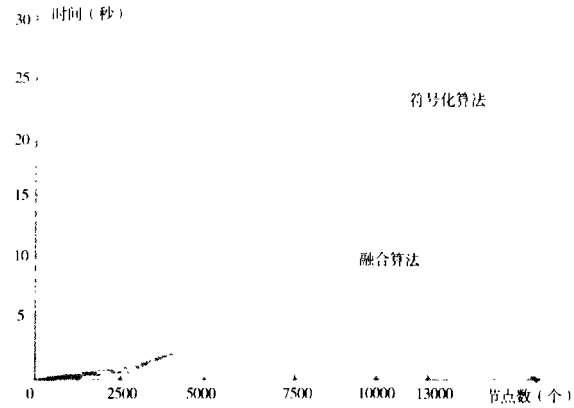

图 4 图平均度数为 100

Fig. 4 Average degree is 100

3 有权图的 BDD表示及算法设计

3.1 有权图的 BDD表示

本文假设图的权值为正整数。如果图的权值为正实数，则可以以适当方式转化为整数进行处理。有权图的 BDD表示的基本思想是把整个图转变为所有边的权值都为 1的图，如图 5所示，即把有权图转化为无权图，这样就可用 BDD来表示了。有权图能用 BDD表示后，就可设计基于 BDD的有权图算法。

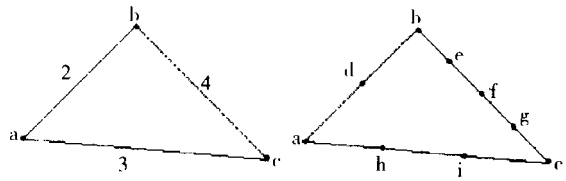

图 5 左图为有权图，右图为对应的无权图

Fig. 5 A weighted graph (left) and its corresponding graph with no weight

3.2 基于 BDD的单源最短路径算法

下面作为一个例子，本文设计基于 BDD表示的单源最短路径算法。设 BDD表示的图为 g，g的节点数为 nodesNum，计算从节点 fromId到节点 endId的最短路径，nodesBDD 为 g的所有节点的 BDD表示数组。算法如下：

（1）设 BDD数组 b[g->nodesNum]，令 i=0，b[0] = nodesBDD[fromId]。

（2）i= i+1; b[i] =Img (b[i-1]);

如果 nodesBDD [endId] 蕴涵 b[i]；则最短路径找到，根据数组 b找出最短路径，结束算法；否则，如果 b[i] 逻辑等值 b[i-1]，则最短路径不存在，结束算法。

（3）转（2）。

3.3 BDD算法与 Dijkstra算法比较

求单源最短路径经典的算法为 Dijkstra最短路径算法[3]，我们编程实现了二算法，在随机生成的图上就运行时间进行比较，所用机器同本文前述。

表 1为随机选取的部分数据，其中，n表示图节点数，m表示边数，w表示图所有边中的最大权值，dTime表示 Dijkstra算法所用时间（单位为 ms），bTime表示 BDD算法所用时间（单位为 ms），ratio表示 bTime与 dTime的比例，Mean ratio表示二算法所用时间的平均比例。用于计算最短路径的节点对是随机产生的。从表中可看出，如果图边的最大权值保持不变，边数和节点数比例相应保持不变，则随着图的节点数增加，BDD算法表现得越来越好，这一点由使用时间的平均比例（Mean ratio）可得出。这也正是 BDD表示的优势所在，规模越大，效果越好。

随着边数的增加，Dijkstra算法的运算时间变化不大，BDD算法的运算时间会缓慢增加。图 6为节点数为 5 000，最大权值为 4的图，随着边数的增加，二算法所用时间的比例的变化情况。从图 6中可看出，BDD算法所用时间随着图边数的增加相对于 Dijkstra算法而增加的很缓慢，当图边数为 20 000时，dTime/bTime等于 0.808；而当边数为 480 000时，dTime/bTime等于 2.21，此时边数增加了 24倍，而时间相应地仅仅增加了 2.73倍，且有进一步变缓的趋势。

随着图权值的增加，Dijkstra算法的运算时间几乎不变，BDD算法的运算时间平均随权值线性增长。图 6为节点数为 2000，边数为 8 000的有权图，随着权值的增加，二算法所用时间的比例的变化情况。从图 6中可看出，BDD算法所用时间随着图权值的增加相对于 Dijkstra算法而平均线性增加。

表 1　随着图节点的增加，二算法所用时间变化情况

Tab. 1　Changes of both algorithms according to the increment of nodes of graph

n=5 000, m=20 000, w=4			n=8 000, m=32 000, w=4			n=10 000, m=40 000, w=4		
Time	bTime	ratio	Time	bTime	ratio	Time	bTime	ratio
1150	870	0.756	1470	1370	0.931	3160	1310	0.414
870	1060	1.218	5400	4080	0.755	1000	600	0.600
470	340	0.723	2690	1680	0.624	6520	3120	0.478
1300	780	0.600	2840	1260	0.443	1380	750	0.543
1470	1120	0.761	5140	3920	0.762	220	240	1.090
1260	1390	1.103	3580	1520	0.424	5590	2480	0.443
1670	1020	0.610	4700	2850	0.606	5230	2560	0.489
1230	770	0.626	3470	2020	0.582	2570	1050	0.408
970	1020	1.051	2800	1240	0.442	3950	1560	0.394
680	430	0.632	2600	1690	0.650	4110	1910	0.464
Mean ratio		0.808	Mean ratio		0.622	Mean ratio		0.532

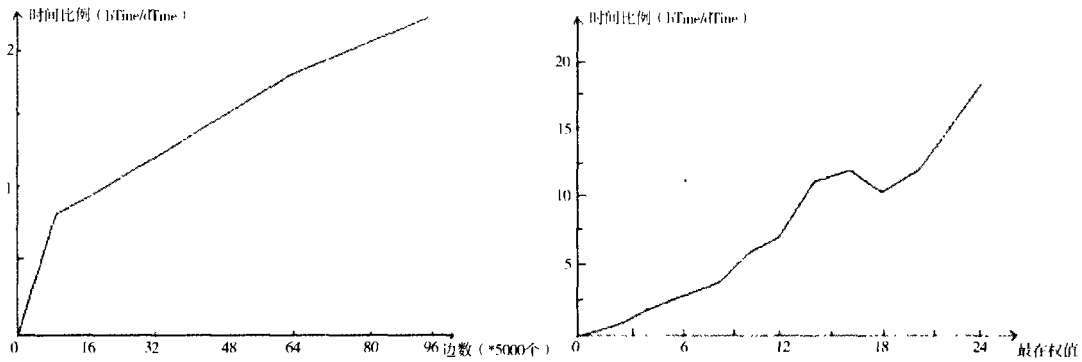

图 6　左图为节点数为 5 000的图（bTime/dTime）随着边数增加的变化情况；右图为节点数为 2 000的图，（bTime/dTime）随着最大权值增加的变化情况

Fig. 6　The left is the changes of bTime/dTime according to the increment of edges of graphs with 5 000 nodes, and the right is the changes of bTime/dTime according to the increment of maximal weight of graphs with 2 000 nodes.

4 相关工作及结束语

在文[13]中提出一种图的符号化表示，即所谓的代数判定图 ADD（Algebraic Decision Diagrams），可用来表示有权图，并利用定义在 ADD上的运算给出了计算最短路径的算法。ADD 实际上是 BDD的一种扩展，但当图权值变化很大时，ADD的空间优势就会失去。本文主要工作是使用 BDD本身来定义和设计图论算法，当图权值变化很大时，仍可有效的压缩空间需求。

本文给出了一种基于二元判决图（BDD）的图的符号化表示方法，尤其是有权图的符号化表示及算法设计。作为例子，我们给出了连通度和最短路径基于 BDD表示的符号化算法，通过实验与常规算法进行了比较，即使对于时间复杂度较低的连通度和最短路径算法，在一定情况下，基于 BDD表示的符号化算法仍然表现出了一定的优势，尤其当图的规模越大时，优势越明显；特别当图规模大到一定程度时，若用常规表示方法，将超过计算机内存，此时由于 BDD的紧凑性，BDD仍然能够在内存中表示，本文用实验证明了这一点。下一步，我们要做的工作是：在图的 BDD表示下，设计其它的图论算法，尤其对于那些时间复杂度很高（最坏情况需要指数时间）的问题，如 Hamilton回路和旅行商（TSP）这类 NP完全问题，以提高此类问题的求解速度。

参考文献：

[1] BRYANT R E. Graph-based algorithms for Boolean function manipulation[J]. IEEE Transactions on Computers, 1986, 8: 677-691.

[2] FEI P Z. Graph, network and their applications[M]. Chengdu: Si Chuan University Press, 1996.

[3] DIJKSTRA E W. A note on two problems in connetion with graphs[J]. Numerische Mathematik, 1959, 1: 269-271.

[4] McMILLAN K L. Symbolic Model Checking[M]. London: Kluwer Academic Publisher, 1993.

[5] CLARKE E, GRUMBERG O, PELED D. Model Checking[M]. Boston: MIT Press, 1999.

[6] LIN H M, ZHANG W H. Model Checking: Theories, Techniques and Applications[J]. Acta Electronica Sinica, 2002, 30(12A): 1907-1912.

[7] NI B, FENG Y L, HUANG T. TBDD Based Dynamic Model Checking[J]. Journal of Software, 1999, 10(10): 1025-1031.

[8] LONG W N, YANG S M, MIN Y H, et al. Variable Ordering Algorithms for OBDD Based on Weight Analysis[J]. Chinese Journal of Computer, 1997, 20(8): 702-710.

[9] BEI J S, BIAN J N, XUE H X, et al. Self-Adaptive Selection Algorithm for OBDD Variable Ordering[J]. Joural of Computer Aided Design and Computer Graphics, 1999, 05(9): 412-416.

[10] SU Kaile, LV Guanfeng, ZHANG Y. Reasoning about Knowledge by Variable Forgetting[C] // In the 9th International Conference on the Principles Knowledge Representation and Reasoning (KR-2004), 2004.

[11] SU Kaile. Model Checking Temporal Logics of Knowledge in Distributed Systems[C] // In Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-2004). 2004.

[12] MEYDEN R van der, SU Kaile. Symbolic Model Checking the Knowledge of the Dining Cryprographers[C] // 17th IEEE Computer Security Foundations Workshop (CSFW-2004), 2004.

[13] BAHAR R I, FROHM E A, GAONA C M, et al. Algebraic Decision Diagrams and their Applications[C] // IEEE /ACM International Conference on CAD, 1993.

Representation of Graph and Its Algorithms Based on BDD

LV Guan-feng, SU Kai-le, LIN Han, LUO Xiang-yu, CHEN Qiang-liang, YUE Wei-ya

(Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, China)

Abstract: A symbolic representation of graph based on ordered binary decision diagram (BDD) is given. The way of how to design algorithms for it is shown. As examples, two symbolic algorithms for connectivity and shortest path problems are given respectively. It is found that the algorithm has obvious advantages especially when the graph becomes larger. When the graph is too huge to be contained in the memory by the routine method, the algorithm here still works due to the compact representation of BDD, which is proved experimentally.

Key words: BDD; symbolic algorithm; connectivity; shortest path