一种基于熵的OBDD变量排序算法*)

赵岭忠　王雪松　古天龙　钱俊彦
(桂林电子科技大学计算机系　桂林 541004)

摘　要　有序二叉决策图(OBDD)是一种有效表示布尔函数的数据结构，其大小依赖于所采用的变量序。熵是定量描述布尔函数中变量重要性的一种方法。基于变量的熵值分析了高质量变量序的特征，给出了一种基于熵的OBDD变量排序算法。实验结果表明：该算法与模拟退火算法和遗传算法结果相当，时间仅为相应算法的80.84%和29.79%。

关键词　有序二叉决策图，变量序，熵

An Entropy-based Algorithm for OBDD Variable Ordering

ZHAO Ling-Zhong　WANG Xue-Song　GU Tian-Long　QIAN Jun-Yan
(Department of Computer Science, Guilin University of Electronic Technology, Guilin 541004)

Abstract　Ordered binary decision diagrams (OBDDs) are a data structure for efficient representation and manipulation of Boolean functions. The size of OBDDs is very sensitive to variable ordering. Based on the characterization of a variable with its entropy, a property of good variable orders is discovered. An entropy-based method for OBDD variable ordering is proposed. Compared with simulated annealing algorithm and genetic algorithm, the algorithm yields comparable OBDDs size with an average of 80.84% and 29.79% runtime, respectively.

Keywords　OBDD, Variable ordering, Entropy of a variable

1　引言

有序二叉决策图(OBDD)是一种有效表示布尔函数的数据结构，最早可追溯到上世纪50年代Lee的二叉决策程序和70年代Akers的二叉决策图(BDD)。Bryant对BDD附加了变量序和简化约束，使之在一定的变量序下表达形式唯一且能够有效地完成多种布尔函数操作[1]。OBDD已成为当前大规模集成电路验证领域广为使用的数据结构。

OBDD表示的空间大小是决定问题能否使用基于OBDD的符号技术求解的关键因素。OBDD的结点数对变量序非常敏感，同一函数在不同变量序下对应OBDD表示的结点数可能具有线性和指数的差别[1]。因此变量排序是OBDD应用中的重要问题。理论研究表明：对任意布尔函数，寻找该函数OBDD表示最小化的最优变量序是NP难问题[2]。计算最优变量序的精确变量排序算法具有指数时间和空间复杂度[3~5]。为此研究人员提出了利用电路结构信息或函数特性的启发式变量排序算法[6~10]和基于相邻变量交换操作逐步改善初始变量序的动态变量排序算法[11,12]，其中Rudell提出的动态变量排序算法sifting已广为应用。此外，多种优化技术被用于变量排序问题的求解，如模拟退火算法(SA)[13]、遗传算法(GA)[14]、演化算法(EA)[8]等。其它变量排序算法包括：通过形成解决问题策略而间接解决该问题的学习型算法、基于样本的变量排序算法、基于概率的变量排序算法[9]等。

从算法的求解质量来看，精确变量排序算法能够求得问题的最优解，GA、SA和分散搜索等算法均可获得质量(以所得OBDD的结点数来衡量)较好的解，但其时间效率低，不适宜求解大型电路变量排序问题。而其它时间效率较高的变量排序算法，如基于电路结构的启发式算法，往往不能得到满意的结果。sifting算法的结果也常常因为搜索的变量序空间有限而较强地依赖于初始变量序的质量。本文在对高质量变量序进行熵分析的基础上给出了一种基于熵的变量排序算法。该算法通过有选择地搜索相对较小但更有希望获得较好结果的变量序空间，降低了算法的运行时间，提高了算法的性能。实验结果显示，该算法与模拟退火算法和遗传算法的结果相当，而时间效率有较大的改善。

2　变量的熵及其扩展

给定布尔函数$f(X)$，其中$X=\{x_1,\cdots,x_n\}$，X中元素的一个排列叫作X的一个变量序。变量序Π中的第i个变量记作$\Pi(i)$。X的所有变量序构成了一个具有$n!$个元素的变量序空间。根据OBDD的唯一性[1]，X的一个变量序Π唯一地确定了布尔函数$f(X)$的OBDD表示，本文把该OBDD及其结点数分别记作$OBDD(f,\Pi)$和$\#OBDD(f,\Pi)$。为了便于描述，下文中把通过算法A得到的变量序称作A变量序，如通过遗传算法获得的变量序称作GA变量序。

熵是定量描述函数中变量重要性的一种方法，与文[10]中的概念一致。其基本思想是用函数f的真值表中对变量x敏感的为真的小项的数目来表示x的重要性。设$M=\{m_1,\cdots,m_k\}$是函数$f(X)$所有为真的小项的集合，其中$X=\{x_1,\cdots,x_n\}$。任给$m_i\in M,x_j\in X$，对m_i中的变量x_j取反后所得小项记作m'_i，如果$m'_i\in M$，则称小项m_i对变量x_j是不敏感的，否则称m_i对变量x_j是敏感的。变量越重要对其敏感的小项的数目越多，如果函数f中所有为真的小项对变量x_j均

*)国家自然科学基金(No. 60563005)和广西科学基金(0542036)。**赵岭忠**　博士生，研究方向：符号计算、形式化技术及其应用等；**王雪松**　硕士生，研究方向：符号计算、形式化方法；**古天龙**　教授，博士生导师，研究方向：实时混杂系统理论、协议验证、形式化方法等；**钱俊彦**　副教授，研究方向：软件工程、模型检验、嵌入式实时系统。

不敏感，那么 x_j 就是冗余变量。f 中对变量 x 敏感的为真小项的数目可由 $h(f_{x=0}\oplus f_{x=1})$ 计算，其中 $h(g)$ 表示函数 g 中为真的小项的数目。函数 f 中变量 x 的熵 $entropy(f,x)$ 定义为：$entropy(f,x)=1-h(f_{x=0}\oplus f_{x=1})/h(f)$。给定函数 f 的 OBDD 表示，$entropy(f,x)$ 的计算可以通过基本的 OBDD 操作 $apply()$ 和 $satisfy\_all()$ 实现。为了使本文给出的算法能够应用于多输出电路或函数的变量排序问题，我们把熵的概念扩展到多输出函数。已知多输出函数 $F=\{f^1,\cdots,f^t\}$，变量 x 在 F 中的熵 $entropy(F,x)$ 定义为

$$entropy(F,x)=1-\sum_{i=1}^{t}h(f^i_{x=0}\oplus f^i_{x=1})/\sum_{i=1}^{t}h(f^i)$$

3 算法思想的提出

基于熵的 OBDD 变量排序算法的设计思想是：以熵为特征值进行变量序特征分析，确定高质量变量序的基本特征；在此基础上使算法仅搜索由该特征所限定的较小的变量序空间，从而达到降低算法运行时间，提高算法性能的目的。与本算法设计思想相近的是基于概率的变量排序算法。该算法以变量的输出概率(output probability) $\delta\{f_i\oplus x\}$ 为变量特征值，并在变量序的特征分析中发现好的变量序中变量的输出概率呈现某种周期性；基于此设计了生成具有该特征的初始变量序的方法和对初始变量序进行优化的 bin-sifting 算法。

输出概率作为变量特征值的不利之处在于该数值仅反映了变量的局部特征，即在多输出函数中对于同一变量 x 选择不同的输出函数 f_i 会得到不同的特征值。文[9]的研究表明，基于输出概率的变量排序算法对输出函数的选择较为敏感，算法性能不稳定。而熵作为表示变量在函数中重要性的量，则不受输出函数选择的影响，反映了变量的全局特征，适于作为变量的特征值。下文首先以熵为特征值对变量序进行分析，获得高质量变量序中具有不同特征值的变量的分布规律，然后利用这些规律指导初始变量序的生成。

4 变量序特征分析

变量序特征分析的结果是设计基于熵的变量排序算法的关键。为此，计算了 LGSynth93[15] 中部分电路变量的熵，并对 SA 和 GA 变量序进行了分析。结果发现在一个较好的变量序中变量大致按照熵值高低相间的规律排列。以 c432 电路为例，图 1 给出了该电路 DFS 变量序(即利用深度优先搜索算法获得的变量序)、sifting 变量序、SA 和 GA 变量序的熵曲线，其中后三种变量序均是以 DFS 变量序为初始变量序获得的。图中横轴表示变量序中的位序，纵轴表示对应位置上变量的熵。图中的水平线表示 c432 电路所有变量熵的均值。从图 1 还可看到：sifting 变量序与 DFS 变量序的熵曲线形状类似，这也从另一个角度验证了 sifting 算法对初始变量序的依赖性。因而我们希望获得一个熵值高低相间的初始变量序，以期获得较好的结果。

图 2 给出了一种生成初始变量序的方法。CreateVariableOrder() 有 3 个参数，entropy_order 是一个包含所有变量的变量序列，其中变量按照熵降序排列；m 是一个正整数，表示每次从 entropy_order 的头部或尾部取 m 或者 $r\times m$ 个变量构成新的变量序，其中 r 是一个大于 0 的实数，表示高熵值变量和低熵值变量相互间隔的程度；N 是变量个数。算法生成的新变量序由 perm 返回。

为了对以上算法生成的初始变量序进行优化，我们在对 sifting 算法进行性能分析的基础上设计了一种综合多种变量寻优顺序的综合 sifting 算法。

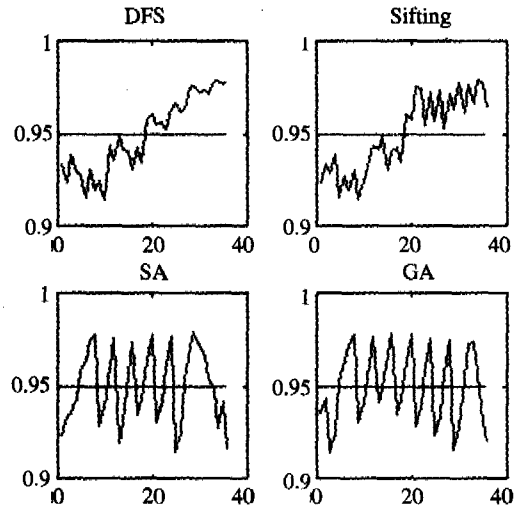

图 1 c432 电路不同变量序的熵曲线

```
Create VariableOrder(entropy_order,m,r)
{  perm=NULL;i=0;j=N/2;
   while((i<N/2)and(j<N))do
   {  if((i<N/6)and(j<5*N/6))
      {  把 entripy_order 中从第 i 个变量起的 m 个变量及
         从第 j 个变量起的 r×m 个变量依次附加在 perm
         尾部;
         i=i+m;j=j+r×m;}
      else if((i<N/2)and (j<N))
      {  把 entropy_order 中从第 i 个变量起的 r×m 个变
         量及从第 j 个变量起的 m 个变量依次附加在 perm
         尾部;
         i=i+r×m;j=j+m;}
      else if(j<N)把 entropy_order 中从第 j 个变量起的
         (N-j)个变量附加在 perm 尾部;
      else if(i<N/2)把 entropy_order 中从第 i 个变量起的
         (N/2-i)个变量附加在 perm 尾部;}
   return(perm);}
```

图 2 基于熵生成初始变量序的算法

5 sifting 和综合 sifting 算法

Sifting 算法的基本思想是：对于给定的初始 OBDD，首先把所有变量按照对应层次上的结点数降序排列，然后依该次序(本文称之为变量寻优顺序)对每一个变量，在保持其它变量相对次序不变的情况下通过相邻变量的交换操作改变该变量的位置，寻找一个使对应 OBDD 结点数最少的最佳位置。所得 OBDD 即为 sifting 算法的输出结果。OBDD 变量排序算法的研究表明，sifting 的性能对初始变量序具有较强的依赖性。而我们对变量寻优顺序的研究则表明，变量寻优顺序对 sifting 算法的性能也具有一定的影响。

除了 sifting 算法中采用了结点数降序(对应节点数较多的变量优先寻找最优位置)，本文提出两种变量寻优顺序：熵降序和熵升序，与之对应的算法分别称作：熵降序 sifting 和熵升序 sifting。在下文中 Rudell 提出的 sifting 算法被称作结点数降序 sifting。我们在 CUDD 软件包[16] 的基础上实现了熵降序 sifting 和熵升序 sifting 算法，并在部分 LGSynth93 电路上以 DFS 变量序为初始变量序执行三种 sifting 算法，比较其结果质量。实验结果如表 1 所示。从表中可看到：三种算法在多数电路上所得 OBDD 结点数各不相同，没有一种算法的性能在所有或多数电路上超过其它算法。

据此，可以给出综合 sifting 算法的基本思想，即综合利用三种不同的变量寻优顺序循环对初始变量序进行优化，直到不能改善结果的质量为止。表 1 的最后一列给出了按照结

点数降序、熵降序和熵升序对 DFS 变量序进行循环优化所得的结果。与其它三种 sifting 算法相比，综合 sifting 算法所得 OBDD 的结点数平均分别减少了 6.23%、5.96%和 9.34%。为此本文采用该算法对 CreateVariableOrder()生成的初始变量序进行优化。综合 sifting 算法的设计思想类似于具有收敛性的 sifting 算法，但实验结果表明该算法更适用于基于熵生成的初始变量序的优化。

表 1 结点数降序 sifting、熵降序 sifting、熵升序 sifting 和综合 sifting 算法性能比较

电路	N1	N2	N3	N4
alu2	172	169	164	158
alu4	675	601	756	575
apex1	1276	1279	1376	1260
apex2	864	554	626	450
apex3	856	856	846	856
apex4	992	903	989	992
apex5	1162	1353	1356	1094
apex6	753	717	752	630
apex7	315	303	300	292
c1355	34234	33218	34930	32106
c1908	7731	8235	7693	7616
c3540	33546	33345	48257	31557
c432	20376	20388	20376	20376
c499	34234	33218	34930	32106
c5315	2619	2574	3067	2335
c880	5999	6112	5979	5946
cps	1073	999	1127	976
ex4p	519	522	502	474
i2	206	338	206	205
i3	133	133	133	133
i6	217	215	220	209
i8	2010	2010	1949	1992
ttt2	107	107	136	107
vda	505	495	507	478
x1	445	483	476	438
x3	627	637	707	627
x4	480	475	485	454

N1、N2、N3 和 N4 分别是所获得 OBDD 的节点数

6 算法描述

基于熵的变量排序算法的主要任务是在具有所希望特征的特定变量序空间中搜索最优变量序。而该变量序空间则由第 4 部分给出的基于熵的初始变量序生成算法所限定。

算法伪代码如图 3 所示。其中，dfs_order 是 DFS 变量序；Max_m 和 Max_ratio 分别是函数 CreateVariableOrder() 的参数 m 和 r 的最大值；step 是控制 r 增长速度的变量。算法中构造 OBDD(f,Π)是基于相邻变量的交换操作实现的，即：对于每一个变量Π(i)($i=0,\cdots,n$)通过若干次的相邻变量交换操作把该变量移至位置 i。

与搜索过程相比，计算变量熵以及对变量按照熵排序可在较短时间内完成。算法运行时间主要由循环次数和循环体的运行时间决定。在算法实现过程中发现：OBDD(f,Π)的结点数有时会产生爆炸式增长，随之导致循环体运行时间激增。为此在构造 OBDD(f,Π)的过程中引入一项限制：如果在移动第 i 个变量的过程中 OBDD 的结点数超出了所设定的范围，或者构造与该变量序对应 OBDD 的总时间超过了某个最大值 MAXTIME，那么算法不再移动剩余尚不满足变量序 Π 的变量，而保持剩余($n-i$)个变量的当前位置不变，直接利用综合 sifting 算法对当前结果进行优化。

```
ReduceBDDbyEntropyBasedMethod(F)
{  计算变量 x_i(i=0,…,n)在 F 中的熵;
   把变量按照熵降序排列,结果保存在 entropy_order 中;
   best_order=dfs_ordr;min_modes=#OBDD(F,dfs_order);
   为 m 和 r 赋初值,且满足 m 为非负整数,r 为正实数。
   while(m<Max_m)do
   {  while(r<Max_ratio)do
      {  Π=Create VariableOrder(entropy_order,m,r);
         构造 OBDD(F,Π);
         调用综合 sifting 算法对 OBDD(F,Π)进行优化并得到新的变量序Π';
         num_n=#OBDD(F,Π');
         if(num_n<min_nodes)
         {  min_nodes=num_n;best_order=Π';}
         r=r+step;}
      m=m+1;}
   return(best_order);}
```

图 3 基于熵的变量排序算法伪码

7 实验结果

我们在 CUDD 软件包的基础上实现了上述算法。本文所有实验均是在 P4 2.6GHz 处理器，内存 256M 的微型计算机上完成的。为了验证算法的性能，对 LGSynth93 中部分电路进行了实验，比较了基于熵的算法和 CUDD 软件包中实现的 SA、GA 的性能。基于熵算法的参数设置情况如下：Max_m=5，Max_ratio=6，限定初始结点的数目不能超过 MIN(5 * dfs_nodes，MAXNODES)，而对构造 OBDD 的总时间不做限制，其中：dfs_nodes 是电路与 DFS 变量序对应 OBDD 表示的结点数，MAXNODES=300000。SA、GA 和 sifting 算法的输入均为 DFS 变量序，其它参数采用 CUDD 软件包中的默认设置。

表 2 基于熵的变量排序算法(EB)和其它算法的性能比较

电路	EB		SA		GA	
	nodes	*time(s)*	*nodes*	*time (s)*	*nodes*	*time (s)*
alu2	154	0.27	154	0.27	154	0.30
alu4	566	0.81	564	1.39	564	1.80
apex1	1246	5.66	1273	7.73	1247	24.30
apex2	339	4.22	321	4.86	331	24.64
apex3	839	4.61	846	6.25	838	21.13
apex4	889	0.73	889	1.28	889	0.77
apex5	1076	10.52	1073	12.94	1044	36.33
apex6	515	9.39	573	7.81	491	24.80
apex7	216	1.91	246	2.69	214	8.70
c1355	26682	913.16	26408	2278.86	26408	11911.34
c1908	5603	72.02	5832	86.53	5534	734.52
c3540	23828	521.84	23836	1338.70	23828	5970.22
c432	1204	156.61	1359	100.59	1215	2168.48
c499	26682	910.24	26408	2288.08	25866	20125.80
c5315	2366	106.27	1799	175.67	1746	214.00
c880	4447	34.28	4082	86.30	4082	1223.76
cps	971	2.48	971	3.67	971	6.59
ex4p	449	3.8	460	3.94	455	12.98
i2	205	35.95	206	46.91	205	183.44
i3	133	2.36	133	7.48	133	29.00
i6	209	1.72	209	2.17	209	9.36
i8	1276	14.44	1297	15.30	1277	51.41
ttt2	107	0.48	107	0.47	107	1.47
vda	478	0.95	484	1.13	478	1.94
x1	425	3.83	411	4.55	409	14.09
x3	502	8.73	522	7.59	503	24.86
x4	331	6.72	368	3.91	340	13.95

其中 nodes 表示对应算法所得 OBDD 的节点数

实验结果见表 2。基于熵的变量排序算法(EB)与 SA 和 GA 性能比较的统计结果如下：在 27 个实验电路上 SB 与 SA 所得结点数之比平均为 99.70%，运行时间之比平均为 80.84%。和 GA 相比，结点数之比和运行时间之比的均值分别为 102.24%和 29.79%，其中在 c1355、c1908、c432、c499、c880

和 i3 上的运行时间之比在 10%以内。

结论 本文把布尔函数中变量熵的概念进行扩展并应用于 OBDD 变量排序算法的设计。熵作为特征值的优点是能够较好地反映变量的全局特征，有利于获得稳定的算法性能。基于熵的变量排序算法通过选择性地搜索更有希望获得较好结果的变量序空间降低了算法运行时间，提高了算法的性能。实验表明：与模拟退火算法和遗传算法相比，基于熵算法的时间效率明显改善，在获得相当结果的情况下，时间仅为相应算法的 80.84%和 29.79%。进一步的工作包括寻找更为系统的变量序特征分析方法，并在此基础上设计更为有效的变量排序算法。

参 考 文 献

1 Bryant R E. Graph-based algorithms for Boolean function manipulation[J]. IEEE Transactions on Computers, 1996, 35(8): 677～691
2 Bollig B, Wegener I. Improving the variable ordering of OBDDs is NP-complete[J]. IEEE Transactions on Computers, 1996, 45(9): 933～1002
3 Friedman S J, Supowit K J. Finding the optimal variable ordering for binary decision diagrams[J]. IEEE Transactions on Computers, 1990, 39(5): 710～713
4 Ebendt R, Günther W, Drechsler R. An Improved Branch and Bound Algorithm for Exact BDD Minimization[J]. IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, 2003, 22(12): 1657～1663
5 Ebendt R, Günther W, Drechsler R. Combining ordered best-first search with branch and bound for exact BDD minimization [C]. ASP-DAC, 2004. 875～878
6 Malik S, Wang A R, Brayton R K, Sangiovanni-Vincentelli A. Logic verification using binary decision diagrams in a logic synthesis Environment[A]. In: Proceedings of International Conference on Computer-aided Design[C], 1988. 6～8
7 龙望宁，杨士元，闵应骅，等. 基于重量分析的 OBDD 变量排序算法[J]. 计算机学报，1997，20(8)：702～710
8 贝劲松，边计年，薛宏熙，等. OBDD 变量排序的自适应选择算法. 计算机辅助设计与图形学学报[J]，1999，11(5)：412～416
9 Thornton M A, Williams J P, Drechsler R, et al. SBDD Variable Reordering Based on Probabilistic and Evolutionary Algorithms [A]. In: Proceedings of IEEE Pacific rim conference[C], 1999. 381～387
10 Jain J, Moundanos D, Bitner J, et al. Efficient variable ordering and partial representation algorithms[A]. In: Proceedings of the 8th International Conference on VLSI Design[C], 1995. 81～86
11 Fujita M, Matunaga Y, Kakuda T. On the variable ordering of binary decision diagrams for the application of multi-level logic synthesis[A]. In: Proceedings of European Design automation Conference[C], 1991. 50～54
12 Rudell R. Dynamic variable ordering for ordered binary decision diagrams[A]. In: Proceedings of International Conference on Computer-aided Design[C], 1993. 42～47
13 Bollig B, Löbbing M, Wegener I. Simulated annealing to improve variable orderings for OBDDs[A]. In: Proceedings of International Workshop on Logic Synthesis[C], 1995
14 Drechsler R, Becker B, Gockel N. A genetic algorithm for variable ordering of OBDDs[J]. IEE Proceedings of Computers and Digital Techniques, 1996, 143(6): 364～368
15 Collaborative Benchmarking Laboratory. 1993 LGSynth Benchmarks. North Carolina State University, Department of Computer Science, 1993
16 Somenzi F. CUDD: CU Decision Diagram Package, Release 2.3.1, 2001

(上接第 206 页)

6 Harry Z, Sheng S. Learning Weighted Naïve Bayes with Accurate Ranking. In: Fourth IEEE International Conference on Data Mining (ICDM'04), 2004. 567～570
7 Pawlak Z. Rough set. International Journal of Computer and Information Sciences, 1982, 11(5): 341～356
8 王国胤. Rough 集理论与知识获取. 西安：西安交通大学出版社，2001. 1～147
9 王国胤，于洪，杨大春. 基于条件信息熵的决策表约简. 计算机学报，2002，25(7)：759～766
10 Wang Guoyin, Zhao Jun, An Jiujiang, Wu Yu. A Comparative Study of Algebra Viewpoint and Information Viewpoint in Attribute Reduction, Fundamenta Informaticae, 2005, 68(3): 289～301
11 Wang G Y, Zheng Z, Wu Y. RIDAS—A Rough Set Based Intelligent Data Analysis System. In: First IEEE International Conference On Machine Learning and Cybernitics(ICMLC2002), 2002
12 Jiang Liangxiao, Zhang Huajie, Cai Zhi hua, Su Jiang. Evolution Naïve Bayes. In: Proceedings of International Symposium on Intelligent Computation and its Application (ISICA 2005), 2005. 344～350
13 Huang Jin, Lu Jingjing, Ling C X. Comparing Naïve Bayes, Decision Tree, and SVM with AUC and Accuracy. In: IEEE International Conference on Data Mining(ICDM2003), 2003. 553～556
14 Huang Jin, Ling C X. Using AUC and Accuracy in Evaluting Learning Algorithms. In: Knowledge and Data Engineering, IEEE Transactions on, 2005, 17(3): 299～310
15 Nadeau C, Bengio Y. Inference for the generalization error. In Advances in Neural Information Processing Systems MIT Press, 1999, 12: 307～313

表 4 实验结果-测试正确率及训练时间表

编号	数据集	NB	AWNB		IWNB		SWNB		HCWNB	
		accuracy(%)	accuracy(%)	time(s)	accuracy(%)	time(s)	accuracy(%)	time(s)	accuracy(%)	time(s)
1	Balance	77.12±2.82	83.12±1.12	1.15±0.07	76.72±3.60	1.32±0.06	81.88±4.43	1.40±0.14	80.72±2.05	1.37±0.10
2	Ballon	90.0±13.69	75.0±17.68	0.12±0.01	80.0±11.18	0.21±0.07	80.0±11.18	0.33±0.07	90.0±13.69	0.25±0.02
3	Breast-Cancer	92.44±2.58	85.0±3.51	0.61±0.05	95.86±2.90	1.03±0.04	95.78±2.32	1.49±0.33	95.96±2.81	1.57±0.15
4	Car	80.5±4.84	88.74±6.55	3.26±0.09	87.7±4.44	3.36±0.14	87.76±5.38	3.66±0.21	84.92±6.04	4.19±0.59
5	Iris	92.32±6.69	94.86±1.43	0.15±0.06	91.32±0.86	0.24±0.03	95.32±3.98	0.33±0.02	94.72±4.19	0.32±0.01
6	Lung-Cancer	40.0±10.46	45.0±6.85	3.19±0.07	55.0±6.85	3.50±0.45	55.0±6.85	5.54±0.28	47.5±5.59	7.35±0.17
7	Mushroom	69.46±4.16	71.76±18.33	38.30±0.42	78.8±1.94	25.81±1.17	80.14±3.15	48.80±0.71	83.64±6.07	53.66±8.67
8	Pima-Indians	72.04±1.99	74.87±2.04	0.41±0.02	74.76±3.46	0.64±0.15	76.26±2.85	0.74±0.13	74.72±1.14	0.75±0.04
9	House-Vote	49.66±7.95	52.64±6.35	1.58±0.02	46.44±4.91	1.55±0.14	53.7±5.04	2.24±0.42	50.32±8.07	2.11±0.85
10	Zoo	87.2±7.53	85.0±3.54	0.65±0.02	92.0±5.70	1.47±0.24	89.6±5.32	1.62±0.17	90.0±7.07	1.54±0.01
11	Glass	64.52±8.75	56.58±7.60	0.63±0.01	67.78±2.01	1.48±0.23	65.84±2.20	1.56±0.20	66.52±6.45	1.62±0.01
12	Wine	81.88±5.89	85.88±1.86	0.75±0.01	86.14±0.74	1.08±0.19	88.62±1.47	1.25±0.11	85.16±4.34	1.14±0.03
13	Hepatitis	80.5±2.67	83.96±3.48	1.01±0.01	84.35±1.10	1.21±0.23	84.26±1.04	1.43±0.29	84.48±1.01	1.79±0.05
14	Letter	63.06±0.52	64.63±2.43	33.10±1.59	65.43±1.69	30.92±1.48	68.83±2.25	58.74±0.52	66.27±4.17	89.43±2.56
15	Chess	70.64±9.10	74.58±3.77	31.62±0.77	75.14±3.97	9.94±0.48	76.4±2.88	34.11±1.09	80.28±2.23	88.48±5.47
	平均值	75.42	76.11	7.77	78.5	5.58	79.96	10.88	79.88	17.04

注：AWNB, IWNB, SWNB, HCWNB 分别表示代数观、信息观、综合代数观和信息观及爬山算法加权朴素贝叶斯方法。