MML Inference of Decision Graphs with Multi-Way Joins

Peter J. Tan and David L. Dowe

School of Computer Science and Software Engineering, Monash University,
Clayton, Vic 3800, Australia
{ptan, dld}@bruce.csse.monash.edu.au

Abstract. A decision tree is a comprehensible representation that has been widely used in many machine learning domains. But in the area of supervised learning, decision trees have their limitations. Two notable problems are those of replication and fragmentation. One way of solving these problems is to introduce decision graphs, a generalization of the decision tree, which address the above problems by allowing for disjunctions, or joins. While various decision graph systems are available, all of these systems impose some forms of restriction on the proposed representations, often leading to either a new redundancy or the original redundancy not being removed. In this paper, we propose an unrestricted representation called the decision graph with multi-way joins, which has improved representative power and is able to use training data efficiently. An algorithm to infer these decision graphs with multi-way joins using the Minimum Message Length (MML) principle is also introduced. On both real-world and artificial data with only discrete attributes (including at least five UCI data-sets), and in terms of both "right"/"wrong" classification accuracy and I.J. Good's logarithm of probability "bit-costing" predictive accuracy, our novel multi-way join decision graph program significantly out-performs both C4.5 and C5.0. Our program also out-performs the Oliver and Wallace binary join decision graph program on the only data-set available for comparison.

Key words:
Machine learning, decision trees, decision graphs, supervised learning, probabilistic prediction, minimum message length, MML, MDL

1 Introduction

In spite of the success of decision tree systems in supervised classification learning, the search for a confirmed improvement of decision trees has remained a continuing topic in the machine learning literature. Two well-known problems from which the decision tree representation suffers have provided incentives for such efforts. The first one is the replication problem which leads to the duplication of subtrees in disjunctive concepts. For example, a decision tree in which

the term (C ∧ D) requires two subtrees to be represented gives an inefficient representation of the proposition (A ∧ B) ∨ (C ∧ D). The effect of the replication problem is that many decision tree learning algorithms require an unnecessarily large amount of data to learn disjunctive functions.

The second problem is the fragmentation problem, which occurs when the data contains attributes with more than 2 values. When decision nodes in a tree are split on high arity attributes (say *arity* ≥ 10), it will quickly fragment the data into many partitions with relatively little data.

Both of these problems increase the size of decision trees and reduce the number of instances in the individual nodes. One way of resolving these problems, particularly when the number of instances in nodes are crucial for inferring the underlying concept, is to use decision graphs to provide an elegant, generalizing solution. Decision graphs can be viewed as generalizations of decision trees and both decision trees and decision graphs have decision nodes and leaves. The feature that distinguishes decision graphs from decision trees is that they may also contain joins (or disjunctions), which are represented by two nodes having a common child. This representation specifies that two subsets have some common properties, and hence can be considered as one subset. As shown on the right hand side of Figure 1, by merging duplicated subtrees with a join operator, the repeated subtrees - from the left hand side of Figure 1 - representing the term (C ∧ D) are united into one subtree.

Fig. 1. Decision tree and Decision graph representations for (A ∧ B) ∨ (C ∧ D)

One attempt as generalizing decision trees was proposed by Mehta et al. [9]. As a refined decision tree system, it allowed multi-way joins but restricted these joins to nodes which have common parents. This improvement does, however, not help with decision tree replication problems such as that of Figure 1.

More recently, Mansour and McAllester [8] introduced a decision graph representation called the branch program. In their system, where decision trees were viewed as a form of boosting, the use of a boosting algorithm guaranteed that the training error declines as $2^{-\beta\sqrt{|T|}}$, where $|T|$ is the size of the decision tree and β is a constant determined by the weak learning hypothesis. Compared to decision trees, whose training error declines as $|T|^{-\beta}$, the branch program showed some promising theoretical results. However, the analysis of the generalization error of the branch program remained open, and thus no comparison with other systems could be made. Together with some earlier works [10, 11, 7, 9], the branch program certainly indicated that decision graphs was a reasonable approach.

The HOODG system, introduced by Kohavi [7], used an Oblivious Read-Once Decision Graph (OODG). OODG was closely related to Order Binary Decision Diagrams(OBDDs). In the HOODG system, variables had to be strictly ordered and decision graphs were generated so that the graph would only test the variables in the specific order. But because the system insisted on a canonical representation, nodes with irrelevant attributes have to be included in the inferred graphs. Kohavi [7] reported significant improvement over decision tree algorithms such as C4.5 and ID3 [13] on artificial data sets generated from disjunctive functions, but the system achieved less success on real-world data sets. The system also did not adequately address noise in data.

Oliver and Wallace's system [10–12] used decision graphs that allowed joins with a pair of nodes. They also presented a MML inference coding scheme and a graph growing algorithm for the system. Our decision graph system builds on the Oliver and Wallace system but eliminates their limitation by allowing for multi-way joins. In this way, our scheme not only generalizes the Oliver-Wallace binary join decision graph but also generalizes the scheme of Mehta et al. [9], which only allowed joins of nodes which have common parents. For a detailed comparison between our multi-way join scheme and the Oliver and Wallace binary join scheme, see section 3.3.

2 Decision Graphs with Multi-way Joins

In this paper, we propose a decision graph system which allows multi-way joins. We use directed acyclic graphs in our system as in both the Oliver and Wallace system [12, 11] and the Kohavi system [7]. In addition to decision nodes and leaf nodes, we introduce join nodes which are merged to have a common child. Join nodes are represented by a lozenge shape in the figures. Although we may represent a join by directly attaching arcs from decision nodes to a child, we have explicitly included them in the figures to clarify our coding scheme. The main idea behind our new decision graph representation is that a decision graph can be decomposed into a sequence of several decision trees. By doing this, we are able to re-use some well-proved decision tree inductive techniques in our system. The details of our algorithm are discussed in section 3.2. Figure 2 illustrates an example of how a decision graph can be decomposed into a sequence of decision trees.

3 MML inference of Decision Graphs

3.1 Bayesianism and MML

The Minimum Message Length (MML) Principle [16, 19, 17] provides a guide for inferring the best model given a set of data. If a set of data is to be transmitted, it can either be transmitted directly (as it is); or alternatively a theory can be inferred from the data, then the set of data is transmitted with the help of the theory. Thus, the transmitted message is composed of the following two parts:

I. the description of the theory, or hypothesis, H

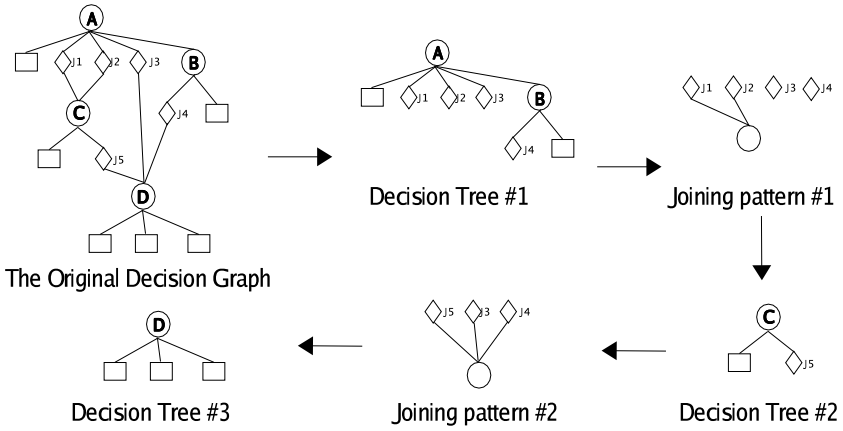

Fig. 2. Decomposing a decision graph into a sequences of decision trees

II. the data, D, explained with help of the theory

From Bayes's rule, we know that Pr(D)Pr(H|D) = Pr(H&D) = Pr(H)Pr(D|H)

So Pr(H|D) $= \frac{Pr(H)Pr(D|H)}{P(D)}$ ..... (1),

where Pr(H&D) is joint probability of D and H, Pr(H) is the prior probability of the hypothesis H, Pr(D) is the marginal probability of the data D, Pr(D|H) is the likelihood function of D given H and Pr(H|D) is the posterior probability of H given observed data D. From (1), we get that

-log Pr(H|D) = - log Pr(D|H) - log Pr(H) + log Pr(D)

To maximize Pr(H|D) is equivalent to minimizing -log Pr(H|D). Because log Pr(D) is a constant, we can ignore it and seek a minimum of - log P(D|H) - log P(H). Thus the hypothesis with the minimum two-part message length can be said to be the model of best fit for the given data. For details of the MML Principle, see [16, 19, 17, 12]. For a comparison between MML and Minimum Description Length (MDL) [15], see, e.g., [17] and other articles in that 1999 special issue of the *Computer Journal*.

3.2 Coding Decision Graphs with Multi-way Joins

Much effort has been put into the development of tree-based classification techniques in recent years. Quinlan and Rivest [14] proposed a method for inferring decision trees using the Minimum Description Length (MDL) Principle [15]. Based on it, Wallace and Patrick [20] presented a refined coding scheme for decision trees using the Minimum Message Length Principle, MML in which they identified and corrected some errors in Quinlan and Rivest's derivation of the message. They also introduced a "Look Ahead" heuristic of arbitrarily many ply for selecting the test attribute at a node.

Oliver et al. presented an inference scheme [12, 11] to construct decision graphs using MML [16, 19, 17]. The machine-learning technique of decision graphs

was successfully applied to the inference of a theory of protein secondary structure from a particular dataset by Dowe et al. [4] (see Section 4.4). The resulting decision graphs provided both an explanation and a prediction method for the problem. However, the Oliver-Wallace coding scheme [12, 11] only allows binary joins. When there are more than two nodes involving in a join, some intermediate nodes have to be created. Immediately below in Section 3.2, we present a refined coding scheme for decision graphs which allows multi-way joins to eliminate such inefficiency.

In this paper, we refine the coding scheme for decision graphs so that a join operation is no longer limited to one pair of nodes. An arbitrary number of nodes can be involved in a join operation and form a common child node. We also refer the reader to the earlier and similar coding scheme of Oliver et al., which only permitted binary joins [12]. To transmit a decision graph now allowing multi-way joins, we use following steps.

1. From the root node of the decision graph, a prefix traversal of the decision tree is performed, treating any join nodes as leaves. Any join nodes that have been transmitted are added to an open list.

2. When step 1 is finished, if this results in any join nodes in the open list, then the combination pattern of the nodes in the open list is described. A combination pattern lists those groups in the open list which combine to have a child. Any groups of nodes in the open list that are involved in a join are deleted from the open list. Add their children to a new node list, in which nodes will become roots of new traversals.

3. If there is any node in the new node list, step 1 is repeated to transmit them.

By decomposing a decision graph into a sequence of decision trees (as in Figure 2), which are transmitted in step 1, we are able to re-use the MML decision tree coding scheme proposed by Wallace and Patrick [20]. Since we treat join nodes as leaves, we implement an adaptive code to describe which leaves are actually join nodes and should be put into the open list.

In step 2 of the above process, a description of the combination pattern of join nodes from the open list needs to be communicated. Firstly, we define the nodes which were added to the open list in the current iterations of transmitting to be "new" nodes, and nodes which were added to the open list in previous iterations then become "old" nodes. We can view the open list as containing N new nodes (from the most recent iteration) and Q old nodes. It should be noted that both N and Q become common knowledge for both sender and receiver before transmission of the combination pattern.

We communicate the combination pattern of join nodes from the open list in four steps, thus four numbers (see points 1-4 below) must be transmitted in the following order.

1. The number of nodes, M, which are children of joins in this iteration. Because there are at least two leaf nodes being involved in any join and because one of these leaf nodes must be a new node, so $1 \leq M \leq min(N, \frac{N+Q}{2})$. Assum-

ing these values to be equally likely, the message length cost of transmitting this number, M, in the message would be $\log(min(N, \frac{N+Q}{2}))$.

2. The number of pending nodes, P, which are not involved in any join and the numbers of nodes $(J_1, J_2, \ldots, J_M)$ in each group of joining leaf nodes from the open list in this iteration.

The task is now equivalent to finding out the number of different solutions to the following equation.

$P + J_1 + J_2 + \ldots + J_M = N + Q$ (where $P \geq 0; J_i \geq 2; i = 1, 2, \ldots, M$)
If the number of valid answers is A and every solution is assumed a priori equally likely, the cost of transmitting those numbers in the message length would be $\log(A)$.

3. The number of "new" nodes , Y, among pending nodes and the number of new nodes, $(X_1, X_2, \ldots, X_M)$ in each group of joining leaf nodes from the open list in this iteration .

Similar to the above step and because there is at least one new node in each joining group, the task is now equivalent to finding out the number of different solutions to the following equation.

$Y + X_1 + X_2 + \ldots + X_M = N$ (where $0 \leq Y \leq P$; $1 \leq X_i \leq J_i$; $i = 1, 2, \ldots, M$)
If the number of valid solutions is B and every solution is assumed a priori equally likely, the cost of transmitting those numbers in the message length would be $\log(B)$.

4. The number of permutations of nodes from the open list in this iteration:
$C = \frac{N!}{Y!X_1!X_2!\ldots X_M!} \frac{Q!}{(P-Y)!(J_1-X_1)!(J_2-X_2)!\ldots(J_M-X_M)!}$
If every permutation is assumed a priori equally likely, the cost of transmitting the number in the message length would be log(C).

After finishing the transmission of the combination pattern, we use an adaptive code to encode the types (i.e., leaf or fork) of nodes in the new node list, followed by new rounds of graph traversal(s). Nodes resulting from join operations can not be join nodes, because if a join operation involving such a node were to happen in a future iteration, the nodes which combined to form this node would have been labeled as "pending" until needed for such a join.

3.3 Comparison with Oliver and Wallace's Decision Graph Program

The difference between our coding scheme and Oliver and Wallace's coding scheme for decision graphs [10–12] has been illustrated in Figure 3. Whenever a multi-way join is required, the Oliver-Wallace coding scheme[10–12] encodes it by proceeding with a series of consecutive binary joins. The result of each such binary join other than the last one is an intermediate node. These intermediate join nodes are redundant (or irrelevant), even though the Oliver-Wallace scheme insists upon encoding them and is thus inefficient. Our scheme is efficient in that it can encode multi-way joins without having to introduce any redundant intermediate nodes. As the example in Figure 3 shows, while the coding scheme with multi-way joins is able to join the four nodes in one step, Oliver and Wallace's scheme has to do it in two steps and two intermediate join nodes J5 and J6 are introduced. The message length caused by coding these intermediate nodes and

various possible ways to form these joins make the scheme less inefficient. For a way of partly removing the inefficiency from the Oliver-Wallace coding scheme, see Section 3.4.

3.4 Unpublished Refinement of Oliver-Wallace Coding Scheme

The Oliver-Wallace coding scheme that we have presented so far [10–12] is inefficient due to the fact that a series of consecutive binary joins is encoded including the order in which those consecutive binary joins occurred, even though that order is in fact irrelevant.

This coding scheme can be rectified and made efficient by accounting for this combinatorial inefficiency. For example, as Figure 3 shows, where four nodes join into one node by three consecutive binary joins, we note that this could have been done in $C_2^4 C_2^3 C_2^2 = 6\text{x}3\text{x}1 = 18$ different ways, and that log(18) can be subtracted from the length of the corresponding (inefficient) message.

It is our understanding that, although such a rectification has not been published, that J.J. Oliver might have since modified the Oliver and Wallace source code to at least partly correct for the above combinatorial inefficiency. However, his program is unavailable.

Such a(n unpublished) correction to the Oliver-Wallace coding scheme would - if correctly implemented - give it a comparable efficiency to the multi-way join coding scheme proposed in our current paper here. The difference between the refined, corrected, Oliver-Wallace scheme (if, indeed, actually and correctly implemented) and our new scheme here would essentially be a choice of Bayesian prior.

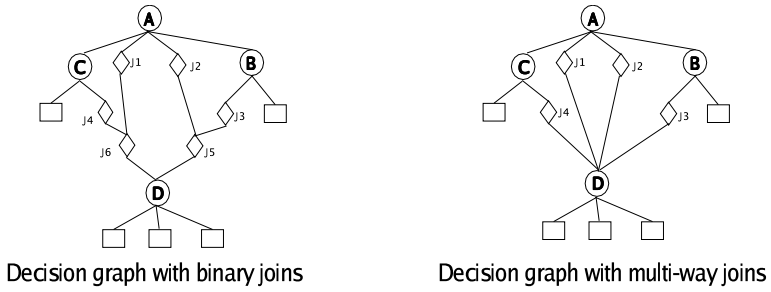

Fig. 3. Different encoding of a function by (left) decision graphs with binary joins and (right) decision graphs with multi-way joins

3.5 Growing a Decision Graph

We begin with a graph having one node, with the root being a leaf. We grow the graph by performing the following procedures iteratively until no further improvement can be achieved.

1. For each leaf L, perform tentative splits on each available attribute in the leaf, and determine the attribute A that will lead to the shortest message length when L is split on A. Record, but do not perform, the alteration (Split L on A) along with its communication saving.

2. For each leaf L, perform tentative joins with other leaves. Record, but do not perform, the alterations (join L_i and L_j; ...; join $L_i, L_j, \ldots, L_k$) along with its communication savings.

3. Sort the alterations from step 1 and step 2 by their communication savings. Choose the alteration (whether from step 1 or from step 2) that has greatest saving.

With help of information which we obtain from step 1, we are able to do heuristic search on potential joins rather than perform possible tentative joins exhaustively. For example, only leaves with communication savings when splitting on an identical attribute can form a join that has a possible message length saving. In the worst case, there will be $2N_aC_2^{|L|}$ tentative joins performed in each iteration, where N_a is the number of attributes available for splitting in leaves, $|L|$ is the number of leaves in the iteration, and $C_2^{|L|} = |L|(|L|-1)/2$.

4 Experiments

Six artificially generated data sets and two real-world data sets were used in the tests. Half of the artificial data sets were generated from disjunctive underlying functions. Most of them were downloaded from the UCI machine learning repository [1] and had been widely tested in other decision tree or decision graph systems [12, 7, 13]. All the data sets we choose have only discrete attributes without missing values. The test results were compared with the well-known classification programs C4.5 and C5 [13], although the Oliver-Wallace binary join decision graph program [10–12] was unavailable to us except for a past result [4] (see section 4.4). In addition to the conventional classification accuracy, a metric called probabilistic costing [6, 5, 2, 3] was implemented for comparison. It is defined as $-\sum_{i=1}^{n} \log(p_i)$, where n is the total number of test data and p_i is the predicted probability of the true class associated with the corresponding data item [6, 5, 2, 3]. This metric can be used to approximate the Kullback-Leibler distance between the true (test) model and the inferred model within a constant.

4.1 Testing with Artificially Generated Data

The experimental results are presented in Table 1, "accuracy" describes the classification accuracy and "pr costing" describes Good's probabilistic costing, or logarithmic 'bit costing' [6, 5, 2, 3]. For the data sets on which 10 10-fold cross-validations were performed, the classification accuracy and probabilistic costing were presented in the form of mean $\pm$ standard deviation, $\mu \pm \sigma$ (where $\sigma^2 = \sum_{i=1}^{N}(x_i - \mu)^2/N$).

4.2 Inferring probabilities for a multinomial distribution

Given an array of occurrences of events of an m-state multinomial distribution $(c_1, c_2, \ldots, c_m)$, the probability of a certain event j can be estimated by (either)

$\hat{p_j} = \frac{c_j + 0.5}{(\sum c_i) + m/2}$ [16, p187 (4), p194 (28), p186 (2)][19][18, p75] or

$\hat{p_j} = \frac{c_j+1}{(\sum c_i)+m}$ [16, p187 (3), p189 (30)], the latter being known as the Laplace estimate and also corresponding (with uniform prior) to both the posterior mean and the minimum expected Kullback-Leibler distance estimator. In our experiments (see Tables 1, 2 and 3), both of the formulas were used to give the probability estimates in our decision graph program while only the latter (+1) was used to give the probability estimates in C5 and C4.5. The first (+0.5) $\hat{p_j}$ estimate was used in the MML multinomial message length calculations, but both $\hat{p_j}$ estimates were used in calculating the decision graph log-prob bit cost.

Table 1. Test Results - Both 'Right'/'Wrong' and Log(Prob) - on Artificial Data-Sets

		D-Graph with M-W joins(+1)	D-Graph with M-W joins(+.5)	C5	C4.5
XD6	accuracy	89.2±1.8%	89.2±1.8%	85.2±2.5%	84.9±2.7%
	pr costing	25.2±2.4bits	27.5±2.7%	30.2±3.3bits	33.6±4.2bits
1st Monk	accuracy	100±0.0%	100±0.0%	75.2±4.3%	75.0±3.9%
	pr costing	9.8±0.0bits	5.0±0.0bits	278.4±12.1bits	285.9±11.3bits
LED-7	accuracy	72.3±3.2%	72.3±3.2%	71.2±3.5%	71.1±3.3%
	pr costing	74.5±7.4bits	72.6±6.2bits	89.7±7.9bits	89.6±7.8bits
LED-24	accuracy	68.5±4.4%	68.5±4.4%	67.8±4.7%	67.8±4.5%
	pr costing	96.4±6.2bits	95.1±6.7%	102.8±7.3bits	103.2±8.0bits
Car	accuracy	91.5±1.7%	91.5±1.7%	91.6±1.9%	91.5±2.4%
	pr costing	44.8±5.6bits	40.5±5.3bits	70.9±6.8bits	72.4±11.9bits
Scale	accuracy	79.2±3.2%	79.2±3.2%	61.9±2.8%	61.4±3.8%
	pr costing	54.6±5.9bits	55.8±6.7bits	83.4±4.7bits	83.3±6.7bits

XD6 data set: The XD6 data set [14, 20, 12] is an artificial set with 10 binary attributes. It was generated so that a division into two categories according to the boolean function of attributes 1 to 9

(A1 ∧ A2 ∧ A3) ∨ (A4 ∧ A5 ∧ A6) ∨ (A7 ∧ A8 ∧ A9)

with 10% noise added to the target attribute. 10 data sets, each of them containing 500 data items, were randomly generated. We then performed 10 individual 10-fold cross-validations on these data sets. Each 10-fold cross-validation consists of 10 tests. In each test, we trained on nine-tenths of the data and tested on the remaining one-tenth. This amounted to 10x10=100 tests.

1st monk's data set: The 1st monk's data set is in the UCI machine learning repository [10, 1], and constructed from the noiseless function

(Jacket_Color = Red) V (Head_Shape = body_Shape)

10 independent tests were performed, with each set of training data consisting of 124 data randomly selected from full data set of size 432. We followed the UCI convention on this data set [1] of using the entire data set as test data.

LED-7 data set and LED-24 data set: The LED-7 data set [1] is an artificially generated data set in the UCI machine learning repository with 7 binary attributes and 10 output classes. Each of the attributes corresponds to one different segment in a Light Emitting Diode that displays the numbers 0 to 9. 10% noise was added to each of the seven input attributes. The only difference between LED-7 and LED-24 is that 17 irrelevant attributes with randomly

generated values were deliberately added in the LED-24 data set. Again, 10 data sets, each of them containing 500 data items, were randomly generated and then an individual 10-fold cross-validation was performed on the each of the data sets - giving a total 10x10=100 tests.

Car Evaluation data set: The car evaluation data set from the UCI repository [1] was generated from an underlying decision tree model. There are 1728 instances with four output classes in the set. Each data item has 6 nominal ordered attributes, which are treated as unordered discrete attributes in our tests because data files from UCI have been set in this format. We performed 10 independent 10-fold cross-validations again on the data set.

Balance scale data set: The balance scale data set from the UCI repository [1] was generated to model psychological experimental results. There are 625 instances with 3 output classes in the set. 10 independent 10-fold cross-validations were again performed on the data set.

4.3 Discussion of above test results on artificial data

As table 1 shows, our decision graph program achieved better or significantly better classification prediction accuracy in most of these test sets (5 out of 6), especially the data sets with a disjunctive underlying function, such as the XD6 and Scale data sets. On the fifth data set (Car), we performed only 0.1% worse. The multi-way join decision graph program also has substantially lower probabilistic bit costing than both C4.5 and C5 on all the test sets. This might best be explained by the fact that decision graphs are more expressive than decision trees and are often able to use the data more efficiently. This should give a shorter Kullback-Leibler distance from the underlying model [17].

4.4 Testing with Real World Data

UCI protein secondary structure database The protein secondary structure database was one of the UCI molecular biology databases [1]. The data contains amino acid chains with a secondary structure specified at each point. Micro-biologists can determine the amino acid chain of a protein, but finding the secondary structures - which are "alpha-helix", "beta-sheet" and "random-coil" - is quite difficult. We constructed decision graphs that predicted the secondary structure at a point in a protein by (following [4] and) using a window of size 7 (centred at the point of interest) of the amino acid chain attributes. Each of these 7 attributes has arity 21. We tested our multi-way join decision graph program, C4.5 and C5 with the default training and test data set downloaded from the UCI repository. As shown in table 2, decision graphs with multi-way joins performed better than C4.5 and C5. In particular, the multi-way join decision graph program achieved at least 2.0% higher classification accuracy and at least 215 bits less in probabilistic costing compared to both C4.5 and C5.

Another protein data set This protein data set was generated from a protein database [4]. We use it because we do not have access to the Oliver-Wallace program [12, 11, 10] and because it is the only data set for which we have the results from that program. We tested C4.5, C5 and our coding scheme

by performing an 8-fold cross-validation on the data set because this was done in the original paper [4]. We also compared our test results with results achieved [4] by the Oliver-Wallace decision graph with binary joins. As shown in Table 3, decision graphs with multi-way joins performed better 6 times out of 8 than each of C4.5, C5 and the Oliver Wallace binary join decision graph program [12, 11, 10]. On average, our scheme achieves 1.6%, 1.5% and 1.2% higher 'right'/'wrong' prediction accuracy respectively than C4.5, C5 and the Oliver-Wallace binary join decision graph program. Our scheme also has a lower probabilistic bit cost score than both C4.5 and C5 on all 8 test sets [4] in Table 3.

Table 2. Test Results of UCI Protein Data Set (From Section 4.4)

	D-Graph with M-W joins(+1)	D-Graph with M-W joins(+.5)	C5	C4.5
accuracy	57.6%	57.6%	55.4%	55.6%
pr costing	4715.2bits	4718.6bits	4935.1bits	4935.5bits

Table 3. Prediction accuracies on another protein data set [4] (from Section 4.4)

Test Set	D-G with Multi-Way Joins	D-G with binary Joins	C5	C4.5	D-G with Multi-Way Joins(+1)	D-G with Multi-Way Joins(+.5)	C5	C4.5
1	53.0%	54.2%	52.4%	52.4%	2296.6bits	2296.7bits	2380.7bits	2380.7bits
2	54.6%	53.3%	53.1%	53.1%	1907.7bits	1907.7bits	1975.2bits	1975.2bits
3	55.9%	51.7%	54.6%	54.6%	2185.4bits	2186.3bits	2241.7bits	2241.7bits
4	58.2%	56.4%	55.8%	55.8%	2066.2bits	2065.9bits	2252.2bits	2252.2bits
5	50.2%	46.8%	43.9%	43.9%	2227.9bits	2228.1bits	2439.2bits	2439.2bits
6	50.7%	49.0%	50.8%	51.0%	2314.8bits	2316.0bits	2362.3bits	2351.3bits
7	53.5%	52.8%	51.4%	50.5%	2218.2bits	2220.3bits	2238.1bits	2295.0bits
8	52.9%	54.6%	54.6%	54.6%	2246.3bits	2247.1bits	2281.0bits	2281.0bits
Avg	53.6%	52.4%	52.1%	52.0%	2182.9bits	2183.5bits	2271.3bits	2277.0bits

5 Conclusion

We have introduced a refined coding scheme for decision graphs which allows multi-way joins. We discussed the use of the Minimum Message Length Principle and the new coding scheme to infer (multi-way join) decision graphs. Our experimental results demonstrated that our refined coding scheme compares favourably with other decision tree inference schemes, namely C4.5, C5 and the Oliver-Wallace binary join decision graph. This pronounced favourable comparison holds both for 'right'/'wrong' prediction accuracy and I.J. Good's logarithm of probability bit costing, and both for artificially generated and real-world data.
We thank Trevor Dix for useful feedback on an earlier draft of this manuscript.

References

1. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/MLRepository.html.

2. D.L. Dowe, G.E. Farr, A.J. Hurst, and K.L. Lentin. Information-theoretic football tipping. In N. de Mestre, editor, *Third Australian Conference on Mathematics and Computers in Sport*, pages 233–241. Bond University, Qld, Australia, 1996.
3. D.L. Dowe and N. Krusel. A decision tree model of bushfire activity. In *Proceedings of 6th Australian joint conference on Artificial intelligence*, pages 287–292, 1993.
4. D.L. Dowe, J.J. Oliver, L. Allison, C.S. Wallace, and T.I. Dix. A Decision Graph Explanation of Protein Secondary Structure Prediction. In *Proceedings of the the Hawaii International Conference on System Science(HICSS) Biotechnology Computing Track*, pages 669–678, 1993.
5. I.J. Good. Rational Decisions. *Journal of the Royal Statistical Society. Series B*, 14:107–114, 1952.
6. I.J. Good. Corroboration, Explanation, Evolving Probability, Simplicity, and a Sharpened Razor. *British Journal Philosophy of Science*, 19:123–143, 1968.
7. Ron Kohavi. Bottom-up induction of oblivious read-once decision graphs: Strengths and limitations. In *National Conference on Artificial Intelligence*, pages 613–618, 1994.
8. Yishay Mansour and David McAllester. Boosting using branching programs. In *Proc. 13th Annual Conference on Comput. Learning Theory*, pages 220–224. Morgan Kaufmann, San Francisco, 2000.
9. Manish Mehta, Jorma Rissanen, and Rakesh Agrawal. MDL-based Decision Tree Pruning. In *The First International Confernce on Knowledge Discovery & Data Mining*, pages 216–221. AAAI Press, 1995.
10. J.J. Oliver. Decision Graphs - An Extension of Decision Trees. In *Proceedings of the Fourth International Workshop on Artificial Intelligence and Statistics*, pages 343–350, 1993. Extended version available as TR 173, Dept. of Computer Science, Monash University, Clayton, Victoria 3168, Australia.
11. J.J. Oliver, D.L. Dowe, and C.S. Wallace. Inferring Decision Graphs Using the Minimum Message Length Principle. In *Proceedings of the 5th Joint Conference on Artificial Intelligence*, pages 361–367. World Scientific, Singapore, 1992.
12. J.J. Oliver and C.S. Wallace. Inferring Decision Graphs. In *Workshop 8 International Join Conference on AI*, Sydney, Australia, August 1991.
13. J.R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann,San Mateo,CA, 1992. The latest version of C5 is available from http://www.rulequest.com.
14. J.R. Quinlan and R. Rivest. Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*, 80:227–248, 1989.
15. J.J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.
16. C.S. Wallace and D.M. Boulton. An Information Measure for Classification. *Computer Journal*, 11:185–194, 1968.
17. C.S. Wallace and D.L. Dowe. Minimum Message Length and Kolmogorov Complexity. In *Computer Journal, Special Issue - Kolmogorov Complexity*, volume 42 of *No 4*, pages 270–283. Oxford University Press, 1999.
18. C.S. Wallace and D.L. Dowe. MML Clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10(1):73–83, Jan 2000.
19. C.S. Wallace and P.R. Freeman. Estimation and Inference by Compact Coding. *Journal of the Royal Statistical Society. Series B*, 49(3):240–265, 1987.
20. C.S Wallace and J.D. Patrick. Coding Decision Trees. *Machine Learning*, 11:7–22, 1993.