An Interactive Framework for Open Queries in Decision Support Systems

Juan A. Fernández del Pozo[1] and Concha Bielza[1]

Decision Analysis Group, Technical University of Madrid,
Campus de Montegancedo, Boadilla del Monte, 28660 Madrid, Spain
{jafernandez,mcbielza}@fi.upm.es

Abstract. We have recently introduced a method for minimising the storage space of huge decision tables faced after solving real-scale decision-making problems under uncertainty [4]. In this paper, the method is combined with a proposal of a query system to answer expert questions about the preferred action, for a given instantiation of decision table attributes. The main difficulty is to accurately answer queries associated with incomplete instantiations. Moreover, the decision tables often only include a subset of the whole problem solution due to computational problems, leading to uncertain responses. Our proposal establishes an automatic and interactive dialogue between the decision support system and the expert to extract information from the expert to reduce uncertainty. Typically, the process involves learning a Bayesian network structure from a relevant part of the decision table and the computation of some interesting conditional probabilities that are revised accordingly.

1 Introduction

In problems of decision-making under uncertainty, the model evaluation outputs are decision tables with the optimal alternatives. In our case, the model is an influence diagram [11], i.e. a directed acyclic graph which has proven to be very useful in this context. Therefore, the evaluation of an influence diagram yields, for every decision time, an associated decision table with the information about which is the best alternative, i.e., the alternative with the maximum expected utility, for every combination of variables (outcomes of random variables and/or other past decisions). The evaluation algorithm determines which these variables are. Let us consider a decision table as a set of attributes (a multidimensional array) that determine the optimal action, policy or response. Then, an essential issue is to minimise its storage space, which typically grows enormously in real-scale problems. Each table may have up to millions of rows and typically, more than twenty columns, and the results of the decision problem, therefore, rely on a combinatorial knowledge representation space.

In [4], we introduce $KBM2L$ lists to address this problem. The main idea stems from how computers manage multidimensional matrices: computer memory stores and manages these matrices as linear arrays, and each position is a

function of the order chosen for the matrix dimensions. KBM2L lists are new list-based structures that optimise this order in the sense of putting equal responses in consecutive positions in order to achieve compact storage.

During usual decision support system (DSS) operation, the expert user will query the system about which is the best recommendation for a given set of attributes. In this paper, we propose a query system based on the KBM2L framework. Section 3 describes different types of queries and elaborates on the difficulties involved. Section 4 addresses the situation of a closed query, with the whole set of attributes instantiated (a record). Section 5 tackles more general and complex queries –open queries–. The procedure combines compact decision tables with learning, information access and information retrieval processes. Section 6 uses a real medical DSS to illustrate all kinds of open queries. The last section is a summary and suggests further research.

2 KBM2L Lists for Knowledge Synthesis

As mentioned above, KBM2L lists store the information of a decision table as a vector, much in the manner computers work. A decision table is a set of attributes. If we assume discrete attribute domains, then an order, natural or conventional, in the values of these domains may also be assumed. The attributes can also be arranged in different orders always maintaining the same information. A *base* is a vector whose elements are the attributes in a specific order. An *index* is a vector whose elements are the values of the attributes of the base, interpreted as the coordinates with respect to the base.

Therefore, the decision table as a multidimensional matrix maps into an array as follows [8]. Given a cell of the table with coordinates $\mathbf{c} = (c_0, c_1, ..., c_n)$, we define $f : \mathcal{R}^{n+1} \rightarrow \mathcal{R}$, such that

$$f(c_0, c_1, ..., c_n) = c_0 \prod_{i=1}^{n} D_i + c_1 \prod_{i=2}^{n} D_i + ... + c_n = q \tag{1}$$

where q is the $\mathbf{c}$-offset with respect to the first element of the table in a given base, D_i denotes the cardinal of the i-th attribute domain ($i = 0, 1, ..., n$), and $\prod_{j=i+1}^{n} D_j$ is called its *weight* w_i. The vector of weights is $\mathbf{w} = (w_0, w_1, ...w_n)$ with $w_k = w_{k+1} D_{k+1}$, $w_n = 1$ and $k = 0, 1, \ldots, n-1$. Their algebraic interpretation is that they are the coefficients that multiply the coordinates in (1), like radix powers in the number value expression. We can use relationship (1) to access all the table values. Without loss of generality, suppose c_i has 0,1,2,...$D_i - 1$ as its possible outcomes and, hence, possible values for q are 0,1,2,...,$w_0 D_0 - 1$.

Rather than looking at the way the cells can be enumerated depending on the base, we now consider their content, namely, the DSS proposals. The list often contains sets of *consecutive* cells leading to the *same* optimal policy, as the optimal decisions present some level of knowledge granularity. The memory savings come from storing only one cell (or equivalently its offset) per set, e.g.

its last cell. This last cell together with the common policy, representing a set of records *(cell,policy)*, is called *item*. The shorter resulting list composed of items is a $KBM2L$ list [4]. It is somewhat related to the way sparse matrices are managed. Each item is denoted as $<offset, policy|$, where the $<$ symbol reflects item offsets increasing monotony and $|$ reflects granularity.

Good KBM2L lists search for bases with few items, grouping identical policies as far as possible into consecutive offsets. We have implemented a genetic and a variable neighbourhood algorithm to guide this search, as well as efficient heuristics for some time-consuming associated processes [5]. A list of minimum length means a table of minimum memory requirements. Moreover, this list also serves as a means of high-level explanation and validation, finding out relationships between groups of attributes and system proposals. [4] illustrates a medical problem. Therefore, this knowledge synthesis is useful from both a physical and a semantic point of view.

3 Queries

The expert decision-maker uses decision tables as a knowledge base (KB). In a typical session, expert and DSS accomplish these tasks: (A) formulate a query in the KB domain; (B) translate it into the KB formalism; (C) implement the response retrieval; (D) build the response efficiently; (E) communicate the response(s) and/or suggest improvements, and wait for user feedback.

As far as tasks (A) and (B) are concerned, there is a wide range of possible query formulations. We propose a simple classification of queries into two groups depending on whether the whole set of attributes is instantiated –*closed* query– or not –*open* query–. A closed query corresponds to a user who issues a specific and defined query, as she knows all the information about the attributes. An open query is more general, since it includes undefined attribute values, because they may be difficult or expensive to obtain or because they are unreliable. This classification is similar to the one given in [9], although [9] deals with GIS (Geographical Information Systems) and focuses more on data efficient updating and access from a physical point of view (merely as a database), than from a logical point of view (as a knowledge base).

Queries are stated as attribute instantiations and, therefore, they are related to the index. A response in terms of the optimal policy is expected. However, an added difficulty is optimal policy ignorance.

Let us explain this point in further detail. The complete evaluation of the decision-making problem may be so costly (in terms of time and memory requirements) that we have to resort to solving a set of subproblems instead, each one as the result of instantiating some random variables [1]. This subproblem set may not be exhaustive, implying unknown optimal policies for some attribute combinations, i.e. those associated with unsolved subproblems. This is common in large decision-making problems. In fact, the optimal KBM2L construction process also operates with unknown policies and subproblems. It firstly (sequentially or in parallel) evaluates all the subproblems, and then the resulting partial

decision tables are sequentially added to the KBM2L list by means of a learning mechanism that optimises the list as new knowledge is entered (i.e., before reading the next partial table). Each stage in the addition process supposes a better item organisation and facilitates future additions [5].

In short, we distinguish not only between closed and open queries, but also between known and unknown or uncertain responses. The different scenarios are analysed in the following sections, thus completing tasks (C)-(E).

4 Closed Queries

The whole set of attributes is instantiated for closed queries. Then, the system only needs to look for the corresponding KBM2L list element and retrieve the optimal policy, which is presented to the user as the response. The mechanism is as follows.

Let $A_0, ...A_n$ be the attributes and $\Omega_0, ..., \Omega_n$ their respective domains. Ω_R denotes policy domain. Let $\mathbf{Q}$ denote a closed query, i.e. $\mathbf{Q} = (a_0, ..., a_n), a_i \in \Omega_i, i = 0, ..., n$. Suppose the optimal KBM2L list has been achieved with respect to base $B = (A_0, ...A_n)$ and $\mathbf{w}$ is its respective weight vector. If this list has h items, then the list is $< q_0, r_0| < q_1, r_1| \cdots < q_h, r_h|$, where $0 \leq q_i \leq w_0 D_0 - 1, q_i < q_{i+1}\, \forall i$ (offsets), $r_i \in \Omega_R, r_i \neq r_{i+1}\, \forall i$ (policies). The response retrieval procedure (task (C) above) consists of projecting $\mathbf{Q}$ into the offset space $\{0, 1, ..., w_0 D_0 - 1\}$ and deriving its policy from the KBM2L list. Namely, if $\langle \cdot, \cdot \rangle$ denotes scalar product, we compute $\langle \mathbf{Q}, \mathbf{w} \rangle = f(\mathbf{Q}) = q$, and whenever $q \in (q_{i-1}, q_i]$ it implies that the response is r_i.

If r_i is unknown, then an efficient solution is to call the influence diagram evaluation and solve the respective subproblem that makes r_i known.

Finally, response r_i displayed by the system to the expert may be completed by asking for an explanation. The expert is not only interested in knowing which is the best recommendation for a certain case, but also in concise, consistent explanations of why it is optimal, hopefully, translated into her own domain language. Explanations are also useful as a sensitivity analysis tool for validating the DSS. Explanations are constructed via two clearly different parts of the indices an item represents. The first part is the *fixed* (constant) part of the indices. The fact that these components take the same value for the respective attributes somehow *explains* why the policy is also the same. Therefore, the set of attributes of the fixed part can be interpreted as the explanation of the policy. The second part, complementary to the first, is the *variable* part: the records do not share the same values and, therefore, the attributes are unimportant information for the item policy [4].

5 Open Queries

We have seen that the expert is an agent interested in the optimal policy for the decision-making problem and she queries the system. Expert and DSS have a dialogue consisting of queries, responses and explanations. For closed queries,

the expert receives definite and accurate responses. For open queries, this task is harder due to expert imprecision. Not all attributes are instantiated. Possible reasons may be the unreliability of some attribute values, missing knowledge, their high extraction cost or simply an interest in making a complex query concerning the whole ranges of (some) attributes. For example, in medical settings, the physician often has access to administrative data like sex, age, etc., but may have no access to (or confidence) attributes like the first treatment received or some test results. Also, she may be interested in asking for all possible patient weight intervals. Thus, an open query is, e.g. $\mathbf{OQ} = (*, a_1, *, ..., a_n)$, where $*$ denotes non-instantiated attribute values.

In principle, the DSS looks up the respective KBM2L list elements and retrieves the optimal policy (or policies) to be presented to the user as the response. Suppose, as above, that the optimal KBM2L list has been achieved with respect to base $B = (A_0, ...A_n)$ and the query is open with respect to attributes i and j, i.e., the query is $\mathbf{OQ} = (a_0, a_1, ..., *_i, ..., *_j, ..., a_n)$. Actually, $\mathbf{OQ}$ is a set of closed queries $\mathbf{Q}_i$, namely,

$$\mathbf{OQ} = \{(a_0, a_1, ..., x, ..., y, ..., a_n) : x \in \Omega_i, y \in \Omega_j\} = \bigcup_{i=1}^{D_i \times D_j} \mathbf{Q}_i.$$

Then, the response retrieval procedure would consist of applying the technique described in Section 4 to each $\mathbf{Q}_i$, computing $\langle \mathbf{Q}_i, \mathbf{w} \rangle = f(\mathbf{Q}_i) = p_i$, to give an offset set $P = \{p_1, p_2, ..., p_{D_i \times D_j}\}$, with the respective responses $S = \{s_1, s_2, ..., s_{D_i \times D_j}\}$.

There exist four possible situations: (i) all s_i are equal and known; (ii) all s_i are equal but unknown; (iii) there are at least two different values among s_i's and they are known; (iv) there are at least two different values among s_i's but some s_i's are unknown. *Situation (i)* implies an accurate response (s_i) and the DSS does not require additional interaction with the expert. The remaining situations involve various possible responses and/or uncertain responses requiring a refinement to be helpful for the expert. It is here that tasks (D) and (E) pointed out in Section 3 play an important role.

The information for the expert comprises two sets P and S, jointly involving simple KB records. They have been extracted from the optimal base. Note that base is the best base for the *whole KB*, both minimising storage requirements and maximising knowledge explanation performance. Notwithstanding, this base may not be useful with respect to the *part* of the KB concerning the open query.

Attribute weight changes depending on its position within a base, and the further to the right the position, the smaller the weight is. We propose moving query open attributes towards positions to the right. The query is the same, but its attribute order implies working in a new base, with open attributes moved towards the smallest weight positions. From a semantic viewpoint, this movement also agrees with the idea of consigning open attributes to less important positions as the query seems to indicate, that is, the expert has not assigned a value to and does not show a significant interest in these attributes. The new base will be called *operative* base, which gives an operative KBM2L. There are several

possible operative bases, all of which are valid. We can choose the base leading to less computational effort when changing the base and transposing the knowledge, after trying a few bases in linear time.

A base change may be interpreted as a change in the query and response points of view. For the DSS, the optimal base represents a good organisation of the whole KB content. For the expert, the operative base provides an organised view of the responses to her open query, as consecutive records. This base will be optimal for explaining the responses. It illustrates how a query may be difficult in one base and easier in another base. It bears a resemblance to human domains, where a query is simple for an expert but hard for non-experts. An expert has a good problem description: a good conceptual model, few relevant facts, and a good information organisation that allows fact analysis and explanation. Indeed, this is why she is an expert.

Now, working on the operative base, new offset set P' will include consecutive p'_i's and we can introduce some rules based on distances in the offset space to make more accurate recommendations to the expert.

Situation (ii) may seem surprising since the DSS is asked for something which is unknown, because the associated subproblems have not been evaluated. Nevertheless, we will try to provide a solution. Specifically, situation (ii) may be solved by predicting that the response is associated with the nearest offset to P' based on the Euclidean distance in the offset space, i.e. in $\{0, 1, ..., w_0 D_0 - 1\}$, as follows. Let p^l, p^u be the minimum and maximum offsets, respectively, included in P'. Suppose q^l is the maximum offset with known policy (say r) that precedes p^l in the operative KBM2L list. Likewise, q^u is the minimum offset with known policy (say s) that follows p^u in the operative KBM2L list. All records that match (p', s), with $p' \in P', s \in S$, belong to the same item in the operative KBM2L list, while, offsets q^l and q^u are located at its adjoining items. Then, we compute $d_1 = |p^l - q^l|$ and $d_2 = |p^u - q^u|$. If $d_1 < d_2$, then the response is r; otherwise the response is s.

Situation (iii) presents different policy values in S. We may give an immediate answer to the expert based on statistics over the policy value distribution (median, mode, etc.). However, more intelligent responses will be preferred. As a first proposal, let us call it Algorithm $\mathcal{A}1$, the DSS asks the expert to instantiate the open attribute further to the left with respect to the optimal base. It will be the most efficient attribute for reducing response uncertainty. That is, it will have the greatest weight among all the open attributes, implying more likelihood of belonging to the fixed part of the item indices, which is what explains a fixed policy. Thus, the further to the left the attribute, the more likely the query is to lead to less responses that are different. If necessary, then, the instantiation of the second further to the left open attribute would be requested, and so on. This approach will fit problems with many attributes but with few open attributes.

For many open attributes, say more than 10, we have enough information to make automatic inferences via a learning process. Thus, we propose focusing once again on the operative KBM2L records of interest and learning the probabilistic relationships among the attributes *and* the policy from these records.

The structure to be learnt is a Bayesian network (BN) (see e.g. [10]), as it has a clear semantics for performing many inference tasks necessary for intelligent systems (diagnosis, explanation, learning...). Then, the resulting structure will provide a basis for starting a DSS/expert dialogue to lend insight into the problem and refine the query in the light of new evidence until the response satisfies the expert.

For the sake of simplicity, let $\mathbf{X} \subset \mathcal{R}^{n_1}, \mathbf{Y} \subset \mathcal{R}^{n_2}, n_1+n_2 = n$ denote, respectively, instantiated and non-instantiated attributes of the query. Our Algorithm $\mathcal{A}2$ follows the steps:

S0. Initialise $\mathbf{X}^0 = \mathbf{X}, \mathbf{Y}^0 = \mathbf{Y}$
S1. DSS extracts data (records) matching $\mathbf{X}^0$ from the operative KBM2L
S2. DSS learns a BN from data (structure and conditional probabilities)
S3. DSS computes $P(R = r|\mathbf{X}^0)$, $\forall r \in \Omega_R$ on the BN. Expert fixes a decision criterion, usually a distribution mode, to choose among r's. Let m^0 be this value. It will be evidence to be propagated through the network.
S4. DSS computes $P(Y_j = y|R = m^0, \mathbf{X}^0)$, $\forall j = 1, ..., n_2, \forall y \in \Omega_{Y_j}$ on the BN. Expert fixes a decision criterion, usually minimum variance, to choose among Y_j's. Let $\widetilde{\mathbf{Y}}^0$ be the resulting vector of Y_j's, with coordinates given by expert instantiations, like e.g., the Y_j mode.
S5. Extend vector $\mathbf{X}^0$ as $\mathbf{X}^1 = \mathbf{X}^0 \cup \widetilde{\mathbf{Y}}^0$. Set $\mathbf{Y}^1 = \mathbf{Y}^0 \setminus \widetilde{\mathbf{Y}}^0$.

Steps *S3* and *S4* are repeated until the expert is satisfied or $\mathbf{Y}^j$ has few components and Algorithm $\mathcal{A}1$ is called to continue. If the algorithm stops at $\mathbf{X}^j$, then m^j is the response returned. The expert can always revise decisions made at *S3* and *S4* of the last iteration whenever she does not agree with the current outputs. Moreover, the DSS will be on the watch for and warn the expert about probabilities conditioned to impossible events (registered in the DSS).

A BN (see a review in [2]) is learned in Step *S2* using a structure learning algorithm, where the K2 algorithm [3] is the standard. It works on quite reasonable data sizes, as in the context we propose. Indeed, we may even have quite large sizes, requiring a sample to be used to be computationally tractable.

Expert decision criteria at Steps *S3* and *S4* might be different. They are expert choices. With the suggested criteria: (a) at Step *S3*, we choose the most likely response given the instantiated attribute set for the query; and (b) at Step *S4*, we choose the attribute(s) in which we have more confidence or the attribute(s) wiggling less than a fixed threshold. Then, they are instantiated in the query as their mode, giving rise to a new, more accurate, query. Posterior steps allow continuous probability updating.

The DSS talks via the BN and its probabilities, giving, firstly, information about the response and, secondly, about the likelihood of each open attribute, given the response chosen by the expert. This support for both responses and queries allows the expert to re-state and improve her query until it is more defined, leading to more accurate answers. This is task (E) mentioned above.

Situation (iv) also presents different policy values in S, some of which, however, may be unknown. Unknown policies are the result of having a system that

can legitimately be termed knowledge based, due to its significant size [6], with the impossibility of being completely solved. The expert plays an important role in deciding which part of the problem should be solved, suggesting which is the core she is interested in. As a consequence, her queries are likely to bear on this core, thus having a known response. Thus, situation (iv) can be solved like situation (iii), trying to avoid unknown policies via, e.g. Algorithm $\mathcal{A}1$.

6 Example from a Medical DSS

Open queries and the respective responses and explanations are the basic dialogue elements in the expert/DSS interaction. We now describe how the dialogue is developed in typical sessions. Our decision-making problem concerns a doctor who has to make a decision about a patient admission and two possible therapy actions $[4, 1]$, i.e. three alternatives: r_0: no admission, r_1: 12-hours observation, r_2: 12-hours phototherapy. The decision table has 12 attributes, see Table 1, and $82,944$ records. The optimal base is $B = (A_0, A_1, ..., A_{11})$, with respective weights $\mathbf{w} = (27648, 13824, 6912, 3456, 1152, 576, 192, 64, 16, 8, 4, 1)$. Our KB has $1,656$ items distributed over $27,736$ known records and $55,208$ unknown records. Each query will be coded according to the optimal base order.

Table 1. Attributes and domains

Attributes	Domains
Concentration of Bilirubin(A_0)	Normal(0), Pathological(1), VeryPathol.(2)
Child's(A_1) & Mother's(A_9) Rh	Negative(0), Positive(1) Factor
Primiparous?(A_2)	Primiparous(0), Multiparous(1)
Delivery with Instruments?(A_3)	Natural(0), Instrumental(1)
MotherAge(A_4)	15-18(0), 19-35(1), >35(2)
Child(A_5) & Mother(A_{10}) Coombs'	Negative(0), Positive(1) Test
5MinApgarTest(A_6)	0-3(0), 4-7(1), 8-10(2) level
Concentration of Hemoglobin(A_7)	Normal(0), Pathological(1), VeryPathol.(2)
BirthWeight(A_8)	0.5-1.0(0), 1.0-1.5(1), 1.5-2.5(2), >2.5(3) kg
Jaundice(A_{11}) Yellow(Y) skin	Normal(0), Y.(1), Y.-Feet(2), Pumpkin- Y.(3)

Situation (i): doctor queries (0,0,0,0,0,0,1,0,*,*,*,*), with four open attributes on the right-hand side, thus coinciding the optimal and operative bases. The response is that r_1 is the optimal policy. Its explanation is the index fixed part of the corresponding item, that is, A_0 is 0,..., A_7 is 0, see Table 2.

Situation (ii): doctor queries (0,0,0,0,0,1,2,0,3,*,*,*). In this case, the response and its explanation are not available (NA) as the query covers only unknown responses, see Table 2. Then, the closest offsets with known responses are found. It follows that $q^l = 447, q^u = 1151$, both with policy r_1. Hence, $d_1 = 561, d_2 = 128$, and the response is r_1. q^u is located at an item with 64

records, from offset 1151 to offset 1215. The index fixed part of this item is $(0, 1, 2, 3, 4, 5, 6, 7)$, with values $(0, 0, 0, 0, 1, 0, 0, 0)$, which is the r_1 explanation.

Table 2. Query details for situations (i), (ii), (iii) and (iv)

	Situation (i)	Situation (ii)
open query	(0,0,0,0,0,0,1,0,*,*,*,*)	(0,0,0,0,0,1,2,0,3,*,*,*)
min index	(0,0,0,0,0,0,1,0,0,0,0,0)	(0,0,0,0,0,1,2,0,3,0,0,0)
max index	(0,0,0,0,0,0,1,0,3,1,1,3)	(0,0,0,0,0,1,2,0,3,1,1,3)
p^l, p^u offsets	192, 255	1008, 1023
fixed part	(0,...,7)	NA explanation
response	r_1	unknown
	Situation (iii)	Situation (iv)
open query	(0,0,1,0,1,1,*,*,*,*,*,*)	(0,0,0,0,0,0,0,*,*,*,*,*)
min index	(0,0,1,0,1,1,0,0,0,0,0,0)	(0,0,0,0,0,0,0,0,0,0,0,0)
max index	(0,0,1,0,1,1,2,2,3,1,1,3)	(0,0,0,0,0,0,0,2,3,1,1,3)
p^l, p^u offsets	8640, 9215	0, 191
fixed part	(0,...,3) for r_1	(0,...,7) for r_1
	(0,...,4) for r_2	NA explanation for unknown
response	r_1 or r_2	r_1 or unknown

Situation (iii): doctor queries (0,0,1,0,1,1,*,*,*,*,*,*), with six open attributes, see Table 2. There are two possible responses, each one with its own explanation. Algorithms $\mathcal{A}1$ or $\mathcal{A}2$ could be called.

For $\mathcal{A}1$, the DSS suggests the doctor to instantiate one of the six open attributes, being A_6 an efficient attribute to start with, as it is further to the left. If it is not possible, doctor would be asked for $A_7, \ldots, A_{11}$ values, in this order, trying to minimise the current alternative set size. In this case, doctor says A_6 is 0, and the DSS answers r_2 is the optimal policy. With only one policy, the session ends. The explanation is (0,0,1,0,1), see Table 2.

Situation (iv): doctor queries (0,0,0,0,0,0,0,*,*,*,*,*). The possible responses are r_1 or unknown, see Table 2. To apply $\mathcal{A}1$, the requested attribute order would be $A_7, \ldots, A_{11}$. Doctor says A_7 is 0, DSS answers r_1, and the session ends. The explanation is (0,0,0,0,0,0,0,0), see Table 2.

We only illustrate Algorithm $\mathcal{A}2$ for situation (iii), since the record set involved by the open query in situation (iv) is too small so as to learn a BN. Situation (iii) query involves more than 500 records. The K2 algorithm was embedded in our main program that manages KBM2L lists. Probability calculi were carried out with Hugin Expert Software [7], once our output was exported to Hugin input format. The BN gives probabilities $0, 0.014, 0.986$ to r_0, r_1 and r_2, respectively, given the data. Doctor chooses the mode, i.e. $m^0 = r_2$, with explanation (0,0,1,0,1) as above. The session ends as doctor is satisfied, not requiring the $\widetilde{\mathbf{Y}}^0$ computation.

7 Conclusions and Further Research

One of the most important DSS facilities is to answer user queries. We have proposed a query system based on the KBM2L framework. Information is efficiently accessed and retrieved to construct the response. The optimal and operative bases allow the records involved in a query to be organised from different perspectives. General queries leading to imprecise responses are addressed via an attributes/policy relationship learning process, where the interaction with the expert is required to arrive at a satisfactory response, with lower uncertainty. Our approach provides the KB definite exploitation for the DSS since queries, responses and explanations are properly performed.

Despite the power of our iterative scheme of progressive knowledge elicitation, future research might focus on allowing queries with constrained rather than non-instantiated attributes, covering initial beliefs about the attributes. Also, more effort could be employed in determining good operative bases if there is more than one. Two criteria could be: minimum computational effort to obtain the new KBM2L and minimum item fragmentation. Finally, rather than directly allowing the expert to choose a decision criterion in Algorithm $\mathcal{A}2$, we could previously implement a search within the tree of possible sessions, i.e. possible r-y-r-$y \cdots$ (response r and instantiated attributes y) sequences. This would filter possibilities that will not satisfy expert expectations, facilitating her choices.

References

1. Bielza, C., Gómez, M., Ríos-Insua, S., Fdez del Pozo, J.A.: Structural, Elicitation and Computational Issues Faced when Solving Complex Decision Making Problems with Influence Diagrams. Comp. & Oper. Res. **27** (2000) 725-740
2. Buntine, W.L.: A Guide to the Literature on Learning Probabilistic Networks from Data. IEEE Trans. on Knowledge and Data Engin. **8** (1996) 195-210
3. Cooper, G.F., Herskovits, E.: A Bayesian Method for the Induction of Probabilistic Networks from Data. Machine Learning **9** (1992) 309-347
4. Fernández Pozo, J.A., Bielza, C., Gómez, M.: Knowledge Organisation in a Neonatal Jaundice DSS. In: Crespo, J., Maojo, V., Martín, F. (eds.): Medical Data Analysis. Lecture Notes in Computer Science, Vol. 2199. Springer, Berlin (2001) 88-94
5. Fernández del Pozo, J.A., Bielza, C., Gómez, M.: Heuristics for Multivariate Knowledge Synthesis. Technical Report #3-UPM. (2002)
6. Henrion, M., Breese, J.S., Horvitz, E.J.: Decision Analysis and Expert Systems. Artif. Intell. Magazine **12** (1991) 64-91
7. Hugin Expert Software: `http://www.hugin.com`
8. Knuth, D.E.: The Art of Computer Programming, Vol. 1: Fundamental Algorithms. Addison-Wesley, Reading (1968)
9. Martinez, C., Panholzer, A., Prodinger, H.: Partial Match Queries in Relaxed Multidimensional Search Trees. Algorithmica **29** (2001) 181-204
10. Pearl, J.: Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Mateo CA (1988)
11. Shachter, R.D.: Evaluating Influence Diagrams. Oper. Res. **34** (1986) 871-882