Knowledge Synthesis on Multidimensional Matrices

Article · January 2001

CITATIONS
0

READS
62

3 authors:

Juan A. Fernández del Pozo
Universidad Politécnica de Madrid
24 PUBLICATIONS **175** CITATIONS
SEE PROFILE

Concha Bielza
Universidad Politécnica de Madrid
369 PUBLICATIONS **6,472** CITATIONS
SEE PROFILE

Manuel Gómez-Olmedo
University of Granada
71 PUBLICATIONS **576** CITATIONS
SEE PROFILE

Knowledge Synthesis on Multidimensional Matrices

J.A. Fernández del Pozo[1], C. Bielza[1], M. Gómez[2]

[1] Decision Analysis Group, Technical University of Madrid, Spain

[2] Computer Science Department, University of Jaén, Spain

{jafernandez,mcbielza}@fi.upm.es, mgomez@ujaen.es

Abstract— **The tables containing the optimal decisions obtained when solving real decision-making problems under uncertainty are often extremely large. This raises problems related not only to the storage and management of so much information, but also to the use of these tables for knowledge retrieval and reasoning explanation purposes. In this paper, we propose turning the tables into minimum storage multidimensional matrices. Computers manage multidimensional matrices as lists, where each position is a function of the order chosen (or base) for the matrix dimensions. The optimal list includes the same knowledge as the original list, but it is compacted, which is very valuable for explaining expert reasoning. Thus, evolutionary computation is required to minimise the number of list entries. The process needs a learning procedure because of its complexity. We illustrate the ideas using our decision support system *IctNeo* [1] for neonatal management, outputting excellent results. The methodology is so general that it also applies to any table considered as a knowledge base.**

Keywords— **Decision Support Systems, Decision Analysis, Influence Diagram, Decision Table, Multidimensional Matrix, KBM2L List, Combinatorial Optimisation, Explanation**

I. Introduction

THE construction of decision analytic models and their implementation as Decision Support Systems (DSSs) involves processes of evaluation, analysis, validation, operation and maintenance. Therefore, the DSSs demanded nowadays are very complex knowledge-based systems. Some important issues that cause this complexity are:

1. The complicated structure of the problem representation, e.g., Bayesian networks [9], influence diagrams [12], with many interrelated elements and constraints. This structure should be able to handle asymmetries and typical protocols (decision sequences) that are beyond the standard models, see [6].
2. The high degree of mathematical formalisation [7]: probability distributions to model uncertainty, utility or value functions to model decision maker preferences, logic, etc., which may lead to a lot of parameters (with uncertainty again in a Bayesian framework).
3. Solving a decision-making problem is a NP-hard problem [3], and its exact evaluation is computationally very complex.
4. Although the inputs are decomposed into small parts (variables, decision stages, joint probabilities factorised in conditional probabilities and additive or multiplicative utility functions), the results rely on a combinatorial knowledge representation space. Specifically, the tables containing the optimal decisions are exponential in terms of number of attributes.
5. Time modelling is also difficult and its forward propagation through the inheritance of direct influences from the past, limits the scope and resolution of its representation.

Moreover, DSSs are considered the human user oriented knowledge-based systems par excellence. They should have an interface and communication close to the user domain and often have to give responses in real time.

The user is actively involved not only during the system construction and use but also during its validation. He/she will demand a system that accepts queries and makes proposals or suggestions in his/her own language. But this is not enough. The user can accept/reject a proposal but it will be better if he/she receives a good, reasoned, consistent and structured explanation. DSSs should provide clear, concise and complete explanations that translate the mechanism of reasoning into the user domain to justify the decisions proposed.

However, the construction of a system providing good explanations is very difficult because [6]:

1. The explanation should be presented from all the possible points of view in a structured and hierarchical way.
2. The explanation models for users and for analysts should be at different levels. Therefore, it should leave the internal aspects of the reasoning model and the inference engine at a second level.
3. It should only use knowledge of the user domain.
4. Explanations should be as general as possible and should emphasise the evidence of both the presence of arguments in favour of the proposal and the absence of arguments against the proposal.
5. The explanation has weak points as a consequence of the uncertainty and the subjectivity inherent to the decision-making process. They should be shown.

In short, the explanation should give a description of why the proposed decision is optimal and new insights into the problem solution. Despite these difficulties, we show here how they can be addressed resulting in useful systems for real problems.

The paper is organised as follows. Section 2 demonstrates how to implement the system knowledge base (KB) in a real computer system by means of a $KBM2L$ structure, a list of a KB multidimensional matrix. $KBMML \equiv KBM2L$, read as $K.B.M.M.to.L. \sim KBM2L$, which states a multidimensional matrix that stores or represents knowledge and is transformed into a list. We will focus on KBs representing decision tables. We will turn them into $KBM2L$ lists. The KB provides the user with system proposals and should build the respective explanations.

Section 3 discusses the problem of optimising the $KBM2L$ structure to get a minimum storage list. Modern global optimisation techniques like simulated annealing or genetic algorithms, will be required. The resulting list will show the implicit rules of the protocol modelled and evaluated by the system, yielding not only the system proposals but also their respective explanations.

Section 4 introduces a medical problem having the above characteristics and its associated DSS, called *IctNeo*. We examine the results of our methodology in Section 5, applying the ideas to *IctNeo*. The process needs a learning procedure because of its complexity.

Thus, the methodology can be used to implement an explanation procedure for the system and to perform sensitivity analyses, as in [2], of the main parameters of interest. It will add important details of validation, affinity and relevance of the attributes of the original table. Moreover, the tasks of case evaluation, learning and organisation may be accomplished in parallel. Section 6 calls for possible applications of these interesting lines of research in the future.

II. Knowledge base implementation using $KBM2L$ lists. Application to decision tables

A KB is the information system used by a DSS to suggest actions and to offer explanations. If it is to be efficient, the KB should be implemented with structure suited to the problem domain. In the case of $IctNeo$ (see Section 4), the knowledge is represented by an influence diagram [12], and the result of the evaluation is a set of decision tables. These tables link the decisions to the variables that influence them (relevant variables).

In general, a decision table can be considered as a set of attributes, relevant variables, that determine an action or a property. The set of attributes of a table will be called *schema*. In this paper, we try to find the relationships among the attributes, using them to minimise the space needed to store the decision tables and to explain the proposals of the DSS.

A. $KBM2L$ lists

The decision tables grow exponentially with the number of attributes. In *IctNeo*, some of the decision tables exceed the storage capacity of any personal computer and so we try to express their contents by means of another representation, minimising their size. We use $KBM2L$ lists.

We begin with some definitions. The order of the attributes in the original decision tables depends on the deletion sequence followed to solve the influence diagram. A change in this order modifies the position of the variables in the schema, but not the proposals of the DSS. If we impose an order on the components of the schema (attributes), a *base* is a vector whose elements are these attributes. We assume an order, natural or conventional, in the values of every domain. So, an *index* is a vector whose elements are the values of the attributes of the base. The index could be interpreted as the coordinates with respect to the base.

Defining an order in the schema attributes and domains allows us to consider the decision tables as multidimensional matrices (MM). So the content of the table stored in the cell with coordinates $\overrightarrow{c} = (c_0, c_1, ..., c_n)$ will be assigned to the position $MM[c_0, c_1, ..., c_n]$. Indeed, this is the way the evaluation program stores and manages the decision tables. In MMs, the values are ordered by means of the application $f : R^{n+1} \to R$, where $f(c_0, c_1, ..., c_n) =$

$$c_0 * \prod_{i=1}^{n} D_i + c_1 * \prod_{i=2}^{n} D_i + ... + c_{n-1} * D_n + c_n = q \quad (1)$$

provides the offset q of a value with respect to the first element of the table. For the i-th attribute ($i = 0, 1, ..., n$), D_i is the cardinal of its domain and $\prod_{j=i+1}^{n} D_j$ is called its *weight* w_i. $\overrightarrow{w}$ is the vector of weights $(w_o, w_1, ...w_n)$ with $w_n = 1$. The weights are the coefficients that multiply the coordinates in (1). The values of the MM will be stored successively in a computer, where only the memory address of the first one is known. Therefore, we can use this relationship to access their values. The meaning of the weight w_i, from the point of view of memory positions allocation and access, is the number of cells separating two values that only differ one unit in the i-th coordinate, i.e., the values

$$(c_0, c_1, ..., c_i, ..., c_n) \text{ and } (c_0, c_1, ..., c_i + 1, ..., c_n).$$

In short, given a cell $\overrightarrow{c}$ we can compute its memory position and retrieve its value. Conversely, the index of the cell can be built from the position, provided the base (the order of the attributes) is known: if $f_B(c_0, c_1, ..., c_n) = q$, then

$$f_B^{-1}(q) = (c_0, c_1, ..., c_n),$$

which may be computed via integer division and modulus operations.

The basic element of the list is the *item*. An item is made up of adjacent cells for which the DSS proposes the same action. The items represent *grains* of knowledge or sets of cases with the same optimal policy. If the content of the table presents some level of granularity, we can store only one value for each group of cases. If we use another base (another order for the attributes), we have the same knowledge but we change the granularity and, hence, the memory requirements to store the final list of items. The objective is to get a base that minimises the number of items, bringing up the grains of knowledge. This also provides a means of explanation, finding relationships between groups of attributes and the proposal of the DSS.

As an example, let the content of a decision table be represented as

$$< (0, *) < (1, *) < \ldots < (p-1, *) < (p, x) < \ldots$$
$$< (q, x) < (q+1, *) < \ldots < (w_0 * D_0 - 1, *)$$

where, each cell of the table is represented as the pair *(offset, policy)*, w_0 is the weight of the first attribute, D_0 is the cardinal of its domain, and the sign $<$ shows the order

among the memory positions used to store the pair. Let us suppose the cells between positions p and q, where $p < q$ without loss of generality, contain the same policy x, as above.

The $KBM2L$ list for this table will include all the positions between p and q in only one item, saving memory space. If there are no other grains of knowledge, then the $KBML2L$ list for the table will be:

$$< (0,*)| < (1,*)| \ldots < (p-1,*)| < (q,x)| \ldots$$
$$< (q+1,*)| < \ldots < (w_0 * D_0 - 1,*)|.$$

The notation $<$ *(offset, policy)*$|$ reflects two ideas. Firstly, the offsets of the items are strictly increasing and, secondly, it summarises a set of adjacent cells with the same policy and different to the proposal of the next item.

There is another problem to solving large influence diagrams: the complete evaluation of the problem may be too costly (in terms of time and memory requirements). So, we have to instantiate some attributes and to solve the reduced diagram, collecting the partial results at the end, see Section 4.

The procedure used to build the system KB follows the steps:

1. Divide the influence diagram evaluation into a complete set of subproblems. These subproblems may be evaluated sequentially or in parallel.
2. The solution of each subproblem is stored in files. One of these solutions is translated to the $KBM2L$ list. Initially this list is empty, with an item representing the complete absence of knowledge (the value of the item is defined as *unKB*). The organisation of the table (the order of attributes in it) determines the initial base (canonical base).
3. The rest of partial results of the other subproblems are added to the list built in step 2, by means of a learning mechanism. Each stage in the addition process supposes an organisation of items (and thereby of attributes) that facilitates future additions. See Section 2.2 below.
4. Optimisation of the final list. This allows the final exploitation of the DSS and provides a high-level explanation of the relationships among attributes. See Section 3 below.

In practice, these four aspects are closely connected in the system implementation and cooperate in the main objective: to get a useful KB for the DSS.

B. From a decision table to KBM2L lists

The formal conditions of consistency of a $KBM2L$ list, where p_j is the offset of item j, and m_j the policy corresponding to p_j, are:

1. $0 \leq p_j \leq (w_0 * D_0 - 1)$, $\forall$ item j in $KBM2L$ list. All the offsets are non negative and smaller than the maximum $w_0 * D_0 - 1$ (since given the base of the MM, it determines the last offset).
2. The offsets are in strictly increasing order ($p_i < p_{i+1}$, $\forall i$).
3. The information or policy of adjacent contexts is different ($m_i \neq m_{i+1}\, \forall i$). If the policies are identical, the cases are joined into a single item by definition of a $KBM2L$ list.

Let d be the policy that will be assigned in cell $\bar{i}$ of the matrix that correspond to a case of the decision table. Let $\bar{i} = (i_0, i_1, \ldots, i_n)$ and $f(i_0, i_1, \ldots, i_n) = x$ be the respective offset. The initial list is: $< (w_0 * D_0 - 1, unKB)|$.

The software we have developed for the construction of the KB by means of a $KBM2L$ list employs 28 rules for the management of the items of a list. It processes the decision tables and the matrices by synthesising the system KB as a $KBM2L$ list. We now detail how is the $KBM2L$ list of items $(offset, policy)$ before and after applying the rules (learning) that add a new case $< (x, d)|$ to the list.

The first seven rules, see figure 1, consider the initial configuration. The list of items consists of a single element and a new element, $< (x, d)|$, is inserted.

The rules from $R8$ to $R14$, see figure 1, are insertions into the first item or at the end of the list. As an example, we illustrate rule $R14$.

Rules $R15$ to $R28$, see figure 1, are insertions into contexts inside and at the end of the list. As examples, we show rules $R25$ and $R26$.

Fig. 1. $KBM2L$ management rules

R1: $set(x = 0, d = unKB)$ in $< (w_0 * D_0 - 1, unKB)| \rightarrow$ $< (w_0 * D_0 - 1, unKB)|$
R2: $set(x = 0, d \neq unKB)$ in $< (w_0 * D_0 - 1, unKB)| \rightarrow$ $< (0, d)| < (w_0 * D_0 - 1, unKB)|$
R3: $set((x > 0) AND (x < w_0 * D_0 - 1), d = unKB)$ in $< (w_0 * D_0 - 1, unKB)| \rightarrow$ $< (w_0 * D_0 - 1, unKB)|$
R4: $set((x > 0) AND (x < w_0 * D_0 - 1), d \neq unKB)$ in $< (w_0 * D_0 - 1, unKB)| \rightarrow$ $< (x - 1, unKB)| < (x, d)| < (w_0 * D_0 - 1, unKB)|$
R5: $set(x = w_0 * D_0 - 1, d = unKB)$ in $< (w_0 * D_0 - 1, unKB)| \rightarrow$ $< (w_0 * D_0 - 1, unKB)|$
R6: $set(x = w_0 * D_0 - 1, d \neq unKB)$ in $< (w_0 * D_0 - 1, unKB)| \rightarrow$ $< (w_0 * D_0 - 2, unKB)| < (w_0 * D_0 - 1, d)|$
R7: $set((x < 0) OR (x \geq w_0 * D_0))$ in $< (w_0 * D_0 - 1, unKB)| \rightarrow$ $ERROR$
... ...
R14: $set(0 < x < p_1, d \neq m_1)$ in $< (p_1, m_1)| < (p_2, m_2)| < (p_3, m_3)| \rightarrow$ $< (x - 1, m_1)| < (x, d)| < (p_1, m_1)| < (p_2, m_2)| < (p_3, m_3)|$
... ...
R25: $set((p_2 < x < p_3) AND (x - 1 > p_2), d \neq m_3)$ in $< (p_1, m_1)| < (p_2, m_2)| < (p_3, m_3)| \rightarrow$ $< (p_1, m_1)| < (p_2, m_2)| < (x - 1, m_3)| < (x, d)| < (p_3, m_3)|$
R26: $set(x = p_3, d = m_3)$ in $< (p_1, m_1)| < (p_2, m_2)| < (p_3, m_3)| < (p_4, m_4)| \rightarrow$ $< (p_1, m_1)| < (p_2, m_2)| < (p_3, m_3)| < (p_4, m_4)|$

These rules of item management are currently being refined to improve their definitions and performance.

III. $KBM2L$ list optimisation

Once we have built the $KBM2L$ list and we know how to manage its items by following the rules, the list must be optimised, as mentioned in step 4 of the previous section. I.e., we now ask which is the order of the schema of attributes or base that minimises the number of items stored for a given matrix.

The base of the matrix shows a certain space and organisation. The information stored is the same but the space is minimum and the organisation is maximum in some bases –optimal bases–. We will see that it is important not only to decrease the size of memory requirements but also to show aspects like affinity among attributes and relevance/irrelevance with respect to the policy (explanation), as the optimal bases do.

The problem of finding the base with minimum storage space is one of combinatorial optimisation. In tables of δ attributes, we must consider $\delta!$ possible solutions, i.e., all the possible permutations of δ attributes. Finding the optimal base is a NP-hard problem that in addition must solve an exponential problem at each step of the algorithm. The improvement of the solutions cannot be verified in polynomial time if the complete lists are compared and there is a factorial number of possible solutions. We propose the use of modern global optimisation techniques, like simulated annealing and genetic algorithms, to solve the problem. We add heuristics to guide the search for the optimum.

A. Information copy procedure

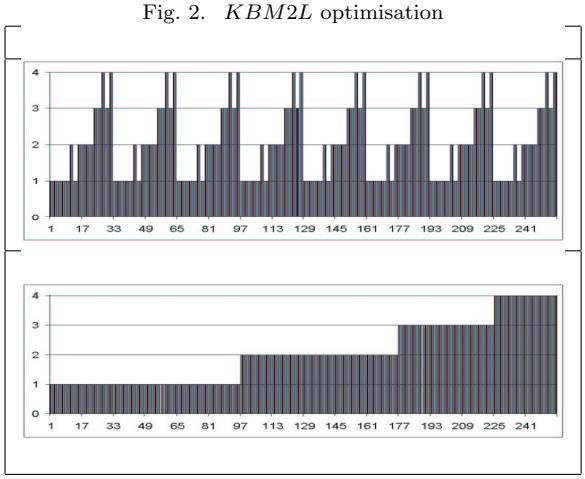
The method for constructing the $KBM2L$ list consists of reading the table inputs with the original initial order $(0, 1, ..., \delta - 1)$ –canonical base– in a structure without information: $KBM2L = (< (w_0 * D_0 - 1), unKB|)$ in (offset, policy)-notation, or $KBM2L = (< ((D_0 - 1, D_1 - 1, \ldots, D_{\delta-1} - 1), unKB)|)$ in (index, policy)-notation. Note that the order of the inputs or cases is not important and, in general, the whole table may not be known, if it is very large. We try to improve the current base generating another base at random. The information is then copied to the new structure. If this new list is more optimal than the old one (i.e., if it requires a shorter list), it is kept and the process iterates to improve it again.

The information copy procedure between $KBM2L$ structures in different bases may require up to $w_0 * D_0$ elements to be copied. If the bases have attributes in common at positions of less weight (those close to w_n), the items have to be copied out following the storage of the $KBM2L$ list. Copying stops whenever the new list fragments the grains or contexts and the new $KBM2L$ list is bigger. In order to get computational savings, we can use the following heuristic: take only a sample of the copy to decide whether the base can be rejected. This means that we do not have to look at the whole copy, which is not an easy task for such massive tables.

Moreover, copying the contents of one structure to another is not so costly from a computational point of view. First, the contents are copied only if they have information, i.e., it is not necessary to copy the inputs marked as unknown, $unKB$. If the matrix tries to gather incomplete information, then there are inputs of unknown policy, which do not belong to the domain of the internal information of the table, symbolised with $unKB$. Second, some contents do not have to be copied. For example, if the policy has three outcomes, the new list may be initialised with the first outcome. Thus, this outcome can be ignored in the copy, and only the other two outcomes have to be copied.

Figures 2 and 3 plot a graphical representation of the $KBM2L$ list that shows how the cases are grouped when the base is changed. This example corresponds to a table in a space of representation with 2^8 cases. In Figure 1, the policy (Y axis), with 4 possible outcomes (1, 2, 3 and 4), is represented against the offset or index in the respective base (X axis). The idea of fragmentation of items with respect to the base is evident when observing both charts. The second one shows the optimal base.

Fig. 2. $KBM2L$ optimisation



The change of base affects the union/fragmentation of the items. And this is the key as we can see in Figure 3 using a storage list spectrum chart we have implemented. $KBM2L$ spectrum shows sets of cases (X axis) which the same optimal decision policy (same color). The lists associated to each spectrum have the same information (256 cases) but the optimal (right-hand) list requires less memory space, see Table I.

B. Parts of an index and union of items

Let I_{inf} and I_{sup} be the vectors of indices associated with the extreme cases of an item of the list. For each one, we now define two clearly different parts from the point of view of this item. The first part is the *fixed* part of the index: its components that are evaluated on concrete values and that are common to all the cases of the item. This is the result of the logical-AND (I_{inf}, I_{sup}). I.e., an attribute $k = v$ is in the explanation if $I_{inf}(i) = I_{sup}(i)$ $\forall i = 0, \ldots, k$. The second part, which is complementary to the first, is the *variable* part: the cases do not share these values and therefore the attributes are irrelevant for the policy associated with the item.

Fig. 3. $KBM2L$ spectrum

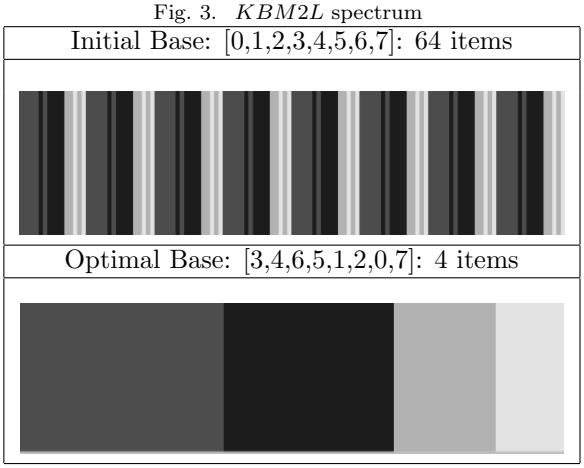


TABLE I
OPTIMAL $KBM2L$ LIST

Initial Base:	[0,1,2,3,4,5,6,7]:	64 items
<((index), policy)|	(lower, upper) offset	cases
<((0, 0, 0, 0, 1, 0, 0, 1), 1)|	(0.0, 9.0)	10.0
<((0, 0, 0, 0, 1, 0, 1, 1), 2)|	(10.0, 11.0)	2.0
<((0, 0, 0, 0, 1, 1, 0, 1), 1)|	(12.0, 13.0)	2.0
<((0, 0, 0, 1, 0, 1, 0, 1), 2)|	(14.0, 21.0)	8.0
<((0, 0, 0, 1, 1, 0, 0, 1), 3)|	(22.0, 25.0)	4.0
<((0, 0, 0, 1, 1, 0, 1, 1), 4)|	(26.0, 27.0)	2.0
...	...	...
Optimal Base:	[3,4,6,5,1,2,0,7]:	4 items
<((index), policy)|	(lower, upper) offset	cases
<((0, 1, 0, 1, 1, 1, 1, 1), 1)|	(0.0, 95.0)	96.0
<((1, 0, 1, 0, 1, 1, 1, 1), 2)|	(96.0, 175.0)	80.0
<((1, 1, 0, 1, 1, 1, 1, 1), 3)|	(176.0, 223.0)	48.0
<((1, 1, 1, 1, 1, 1, 1, 1), 4)|	(224.0, 255.0)	32.0

The above definitions about index parts, have the following implications:
1. An item that is a single case, has its whole index fixed.
2. On the contrary, an item that gathers all or most of the possible cases of a table, may have an empty fixed part. None of the attributes are important in the policy. The description of the item is more complicated for explanation purposes and requires the analysis of the domain of the attributes of greater weight.
3. The analysis of the index as regards the concepts of fixed and variable parts, together with the vectors I_{inf} and I_{sup} means that the set of attributes of the fixed part can be interpreted as the explanation of the policy. The rest of the set is unimportant information. Hence, both optimising the storage of the decision table and finding explanations are to some extent the same problem.

Example. Let us take the matrix in the base $\{c_0, c_1\}$:

$\{c_0, c_1\}$	0	1	2	3	4
0	0	0	0	0	1
1	1	1	1	1	1
2	1	1	1	1	1
3	0	0	1	2	1

Its linear storage in memory may be represented via offsets:

$$\begin{aligned} < & (0,0) < (1,0) < (2,0) < (3,0) < (4,1) \\ < & (5,1) < (6,1) < (7,1) < (8,1) < (9,1) \\ < & (10,1) < (11,1) < (12,1) < (13,1) < (14,1) \\ < & (15,0) < (16,0) < (17,1) < (18,2) < (19,1) \end{aligned}$$

or indices:

$$\begin{aligned} < & ((0,0),0) < ((0,1),0) < ((0,2),0) \\ < & ((0,3),0) < ((0,4),1) < ((1,0),1) \\ < & ((1,1),1) < ((1,2),1) < ((1,3),1) \\ < & ((1,4),1) < ((2,0),1) < ((2,1),1) \\ < & ((2,2),1) < ((2,3),1) < ((2,4),1) \\ < & ((3,0),0) < ((3,1),0) < ((3,2),1) \\ < & ((3,3),2) < ((3,4),1) \end{aligned}$$

The derived $KBM2L$ list has 6 items:

$$\begin{aligned} < & (3,0)| < (14,1)| < (16,0)| \\ < & (17,1)| < (18,2)| < (19,1)| \end{aligned}$$

in offset notation, or

$$\begin{aligned} < & ((0,3),0)| < ((2,4),1)| < ((3,1),0)| \\ < & ((3,2),1)| < ((3,3),2)| < ((3,4),1)| \end{aligned}$$

in indices notation. It should be checked the other base has 4 items more.

As an illustration of the parts of an index, let us take item $< (3,0)|$. It holds $I_{inf} = (c_0 = 0, c_1 = 0)$ and $I_{sup} = (c_0 = 0, c_1 = 3)$. Then, the item contains 4 cases, c_0 is the index fixed part and c_1 the variable part. $i_0 = 0$ explains to some extent the fact that the policy is 0. If we take item $< (19,1)|$, then $I_{inf} = (c_0 = 3, c_1 = 4) = I_{sup}$. Thus, this an item that is a single case and has its whole index fixed. □

Remember that one base is better than another if outputs a smaller $KBM2L$ list. One clearly has to reorganise the matrix in bases that allow the items to be joined, that place equal items adjacently. For example, suppose we have two separate items with offsets p and q and equal policy

$< (p,x)\|$ with index c_1, c_2, c_3, c_4, c_5, where c_1, c_2, c_3 are fixed and c_4, c_5 are variable $\| < (p,x)\| \leq \|c_4\| * \|c_5\|$
$< (q,x)\|$ with index c_1, c_2, c_3, c_4, c_5, where c_1, c_2, c_3, c_4 are fixed and c_5 is variable $\| < (q,x)\| \leq \|c_5\|$

$\| < (p,x)\|$ denotes the number of cases the item includes.

Some of the indices $\{c_1, c_2, c_3\}$ would have to move to a position of lesser weight. Note that if any of the fixed indices cover its whole domain, the respective attribute is irrelevant and both items can be joined. The attribute must be transposed to a position of lesser weight. Thus, suppose $c_j = 0, 1 \, \forall j$ and the situation that follows

$< (p, x)\|$ with index c_1, c_2, c_3, c_4, c_5, where $c_1, c_2, c_3 = 0$ are fixed and c_4, c_5 are variable $\| < (p, x)\| = \|c_4\| * \|c_5\|$
$< (q, x)\|$ with index c_1, c_2, c_3, c_4, c_5, where $c_1 = 0, c_2 = 1, c_3 = 0, c_4 = 0$ are fixed and c_5 is variable $\| < (q, x)\| = \|c_5\|$

Both items are joined and item r is the union of items p and q

$< (r, x)\|$ with new index c_1, c_2, c_3, c_4, c_5 where now $c_1, c_2 = 0$ are fixed and c_3, c_4, c_5 are variable $\| < (r, x)\| = \|c_4\| * \|c_5\| + \|c_5\|$

The initial list is in $(offset, policy)$-notation:

$$< \quad (3, x)| < (7, y)| < (9, x)| < (31, y)|$$

where $p = 3$ and $q = 9$.
And in $(index, policy)$-notation:

$$\begin{aligned} < \quad & ((0, 0, 0, 1, 1), x)| < ((0, 0, 1, 1, 1), y)| \\ < \quad & ((0, 1, 0, 0, 1), x)| < ((1, 1, 1, 1, 1), y)| \end{aligned}$$

The final list after the union of items is, in $(offset, policy)$-notation $< (5, x)| < (31, y)|$, where $r = 5$. In $(index, policy)$-notation, $< ((0, 0, 1, 0, 1), x)| < ((1, 1, 1, 1, 1), y)|$. Note that, the union of "x" items implies the union of "y" items. Thus, the whole list would be, in $(offset, policy)$-notation:

$$\begin{aligned} < \quad & (0, x)| < (1, x)| < (2, x)| < (3, x)| < (4, x)| \\ < \quad & (5, x)| < (6, y)| < (7, y)| < (8, y)| \ldots \\ < \quad & (30, y)| < (31, y)| \end{aligned}$$

And in $(index, policy)$-notation:

$$\begin{aligned} < \quad & ((0, 0, 0, 0, 0), x)| \ldots < ((0, 0, 1, 0, 1), x)| \\ < \quad & ((0, 0, 1, 1, 0), y)| \ldots < ((1, 1, 1, 1, 1), y)| \end{aligned}$$

Note the vectors I_{inf} and I_{sup}. The fixed part of the first item is (c_1, c_2) and the variable part is (c_3, c_4, c_5). For the second item, its fixed part is empty and its variable part is $(c_1, c_2, c_3, c_4, c_5)$.

Table II shows a summary of this operation of union of two items. The initial base is $\{1, 2, 3, 4, 5\}$ and the new base is $\{1, 3, 2, 4, 5\}$.

The complexity of the algorithm lies in the analysis of the fragmented items and in the copy associated with the change of base.

The following subsections describe how the storage is really optimised. Evolutionary computation methods, like simulated annealing [8] and genetic algorithms, are outlined, complemented with some heuristics. All of them cooperate to reach the solution to the problem.

TABLE II
SUMMARY OF THE UNION OF ITEMS p AND q

c_1	c_2	c_3	c_4	c_5	offset	policy
0	0	0	0	0	0	x
0	0	0	0	1	1	x
0	0	0	1	0	2	x
0	0	0	1	1	p=3	x
0	1	0	0	0	8	x
0	1	0	0	1	q=9	x
c_1	c_2	c_3	c_4	c_5	offset	policy
0	0	0	0	0	0	x
0	0	0	0	1	1	x
0	0	0	1	0	2	x
0	0	0	1	1	3	x
0	0	1	0	0	4	x
0	0	1	0	1	r=5	x

C. Global optimisation algorithms

The problem we face is to find a MM base, whose $KBM2L$ list has the minimum number of items. The canonical base has the associated order $(0, 1, 2, \ldots, \delta - 1)$. Thus, there are $\delta!$ bases. That is, it is finite, but huge and cannot be enumerated. We can generate bases at random (using a multistart strategy) and try improvement by means of the change of base. The process of base generation learns how to generate better lists, with a greater probability of union of contexts.

The probability of obtaining the optimal base in a random test is $\frac{1}{\delta!} = p_\delta$, if all attributes are relevant. If we know prior to the study that there are v relevant attributes in all items the above probability is $\frac{v!}{(\delta)!} = p_{\delta - v}$. Note that $p_{\delta - v} > p_\delta$ if $v > 0$.

Let us suppose we are going to move from base B to base B'. Let $L_{kbm-base_*}$ be the initial list length associated with the canonical base, $L_{kbm-base_B}$ the length associated with base B, $L_{kbm-base_{B'}}$ the length associated with base B', and S_B and $S_{B'}$ the lower bounds of the length of the list in their respective bases. If we want a change of base to reduce the size of the $KBM2L$ list, and
$Prob(S_B \leq L_{kbm-base_B} \leq L_{kbm-base_*}) = Prob_B \geq p_\delta$,
$Prob(S_{B'} \leq L_{kbm-base_{B'}} \leq L_{kbm-base_*}) = Prob_{B'} \geq p_\delta$,
$S_B > S_{B'}$
then, we should extract the new base B' from a subset that contains the optimal base and bases that are better than those generated by base B, if we want $Prob_{B'} > Prob_B$ to be held.

Thus, we must generate information relative to the search space that can guide the process. We now propose some heuristics in order to reach a satisfactory solution in a reasonable time.

Heuristic 1. We propose moving through the space of permutations towards elements of Hamming distance $< H$ (H-environments)[1] and compute statistics (minimum, maximum, average item size, etc.) that describe the items of the $KBM2L$ list. With this information, we impose

[1]For a base $\{0, 1, 2, 3\}$, its 2-environment is: $\{\{1, 0, 2, 3\}, \{0, 2, 1, 3\}, \{0, 1, 3, 2\}, \{2, 1, 0, 3\}, \{0, 3, 2, 1\}, \{3, 1, 2, 0\}\}$, which contains six bases.

constraints on the order we generate to prevent item fragmentation and promote unions. The solutions tested are memorised so as not to repeat calculations, and the separation of affine attributes is avoided to a certain extent.

Heuristic 2. Give more opportunity or probability of movement to the corresponding fixed part of the small items (those which represent few cases) towards the zone of the variable part. This is equivalent to generating permutations where some fixed attributes of small items lose weight. If the weight is not suitable[2] for an attribute it causes item fragmentation.

Heuristic 3. As mentioned above, the comparison of lists in different bases should not exhaustively copy the MM content. Thus, stop the copy if a threshold of fragmentation is reached in the new list. However, the movement outstanding items may be key in the union of many items. If various grains of equal policy give rise to one grain others may join as well. The partial copy should combine the copy of cases generated at random and systematically (development of the index). The extreme indices, I_{inf} and I_{sup}, determine the length and position of the items and they are important in the improvement test.

In problems where the size of the set of attributes is very large, it may be necessary to set the weight or position of some attributes a priori in the base and to proceed by learning the subproblem. This is equivalent to fixing the role of an attribute like relevant (high weight) or irrelevant (low weight).

Genetic algorithm. The problem of finding the optimal $KBM2L$ list and the description of its solution is well suited to the use of a genetic algorithm as a possible method of getting solutions. The optimal base is a chain of codes of the attributes of the schema that describes the sequences of nearby attributes (genes), i.e., the characters of the knowledge collected in the table. The items, or rather their descriptions (fixed and variable parts), guide the synthesis of the list. This avoids the random search by building the optimal base from other bases.

The idea of an evolutionary algorithm for this problem stems from the above observation that the index is divided into two parts from the point of view of an item. If we want two items of equal content to become only one item, we should consider permutations constrained to their fixed parts towards the variable part. Several items of the list are joined in a suitable base. This base is better than another in which there is fragmentation of items of the list.

A population of bases can be generated from combinations of $fixed$ and $variable$ parts of the items of common policy. The local permutations on the $fixed$ and $variable$ parts of the index of a set of grains with equal policy do not affect fragmentation. These individuals (bases) are then not tested since they provide no improvement. The schema and the base is the same in all the items of equal policy, the difference lies in the cutoff attribute, i.e., the attribute from index $variable$ part of the base that is adjacent to the index fixed part. All items have their cutoff attribute. One must also consider permutations that interchange attributes of both parts. Crossover operations that take into account these properties are a possibility.

[2]It is suitable for the optimal base.

IV. Example: *IctNeo* system

IctNeo is a complex DSS for neonatal jaundice management, a very common medical problem in which bilirubin accumulates when it is not excreted by the liver at a normal rate. Its first version is already implemented at the Neonatology Service Department of the Gregorio Marañón Hospital in Madrid. The model includes the process of admission, treatment and discharge of a patient. The main objectives are to include a lot of uncertain factors and decisions, define when it is best to order and/or change the treatment, decrease the costs of diagnostic and therapeutic phases, decrease risks due to, e.g., a blood exchange, which is sometimes needed, and take into account the preferences of parents and doctors, see [1]. The hospital hopes to rely on an automated tool for this decision-making problem as an aid to the improvement of jaundice management, getting a better understanding of the problem.

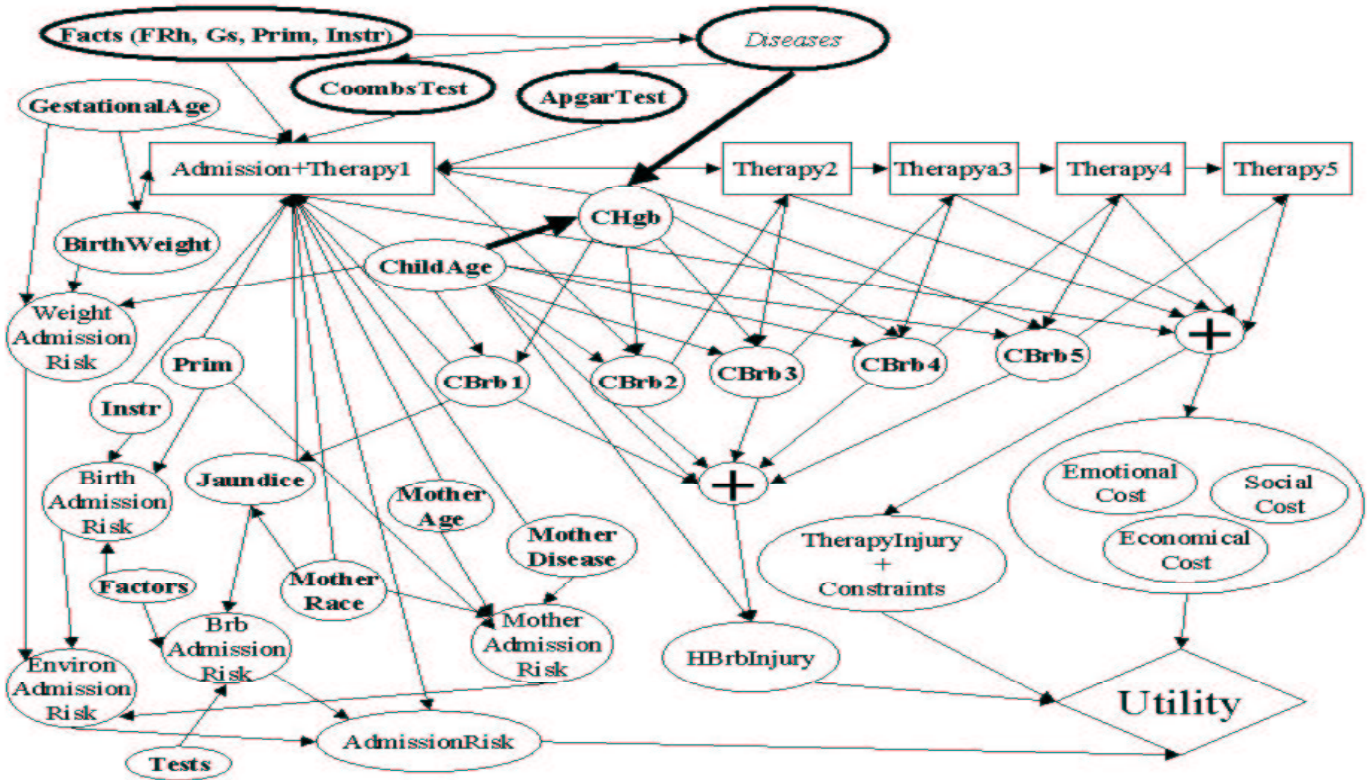
IctNeo represents and solves the problem by means of an influence diagram, see Figure 4, a tool which is becoming more and more popular in Decision Analysis. While conceptually simple, the application of influence diagram methodology may be extremely involved for large problems in practice. The development of the diagram was very complex and time consuming including tasks related to problem structuring and knowledge acquisition (probabilities and utilities), see [1], [5].

The evaluation of the diagram to obtain the optimal policy was even harder. The conventional evaluation [12] provides, at each decision node, a table with the optimal alternative for *each possible combination* of outcomes of the value node predecessors at that moment. This is the knowledge base. Once the system has generated all these tables, it only needs to look for the combination that coincides with the patient under consideration. The sizes of the decision tables are exponential in terms of the number of attributes.

In our case, our large influence diagram entails computational intractability, mainly because the need for storage space (in those tables) grows enormously during the problem-solving process due to node inheritances at chance node removal and arc reversal operations (requirements of approximately 10^{24} memory positions).

IctNeo adds some procedures to the standard evaluation algorithm, see [11]. The system operates with closed cases, comparing its recommendation with the action already taken by doctors concerning the patient in question. Therefore, one way to proceed is to evaluate the diagram by instantiation of evidence on some nodes [4], which amounts to solving the problem for each particular patient once this evidence has been propagated and has updated the influence diagram. Consequently, we have to evaluate as many diagrams as different cases are studied, but the decision tables are smaller.

Nevertheless, the tables are still too big, and they have to



be accessed by the system to propose the treatment stored in the knowledge base. This is the topic of this paper: how to manage the tables and extract the information that is compact enough to easily provide explanations. The *IctNeo* system has motivated our ideas and will serve to illustrate the results.

Currently, as we said above, *IctNeo* is installed at the hospital and operates with minimum requirements (1 megabyte). This version has an influence diagram with five decision nodes. It covers the first 96 hours of a baby's life, which are the most critical ones. The diagram, see Figure 4, has 61 nodes, 137 arcs, 5,586 probability entries, where the biggest table has 748 entries, and an initial table for the value node of size 5,400.

Each evaluation by instantiation takes approximately an hour on a 200-MHz Pentium PC. The maximum size achieved during the process is 10^6 storage positions. The optimal strategy involves five decision tables. It leads to 67 possible full five-stage treatments for a given patient. The first decision depends on nineteen variables, whereas the second stage, depends on eighteen from these nineteen, on the first decision and on another two variables. For illustration purposes, we only show here the second decision, see Table III. The first column indicates the set of attributes, while the second contains their respective domains or instantiated values. The schema of the remaining decisions is analogous to that of the second decision.

An entry in the tables is a whole instance of a case (patient). All the attributes take certain values of their domains. When the case in question is located in the tables, the expected utility $U_{therapy,i}$ of each possible alternative i is shown. For example, see Table IV.

Hence, our problem with five decision stages amounts to having an optimal strategy indicating what to do when making all the decisions for each combination of the variables in the tables. Indeed, we evaluated partially the whole diagram by instantiating the evidence on seventeen nodes. We selected an instance set which represents the real problem for the doctors. Taking into account the cardinal of each attribute domain, the number of instances is

$$|ChildAge| * |ChildRhF| * |MotherRhF| * |ChildAB0F| * |MotherAB0F| * |ApgarT| * |CRhFICT| * |MRhFICT| * |CAB0FICT| * |GestAge| * |BirthWeight| * |Prim| * |Instr| * |J| * |MotherAge| * |MotherRace| * |MotherDisease| = 3 * 2 * 2 * 4 * 4 * 3 * 2 * 2 * 2 * 3 * 4 * 2 * 2 * 4 * 3 * 4 * 2 = 21,233,664 \text{ instances.}$$

The number of entries (combinations) for Therapy2 table is

$$|CBrb1| * |CBrb2| * |Therapy1| * |CHgb| = 3 * 3 * 7 * 3 = 189 \text{ entries.}$$

Therefore, the whole Therapy2 table has $21,233,664 * 189 = 4,013,162,496$ entries. Each entry has seven associated expected utilities. Thus, 4,013,162,496 * 7 = 28,092,137,472 real numbers. Each one needs 10 bytes requiring a total of 280,921,374,720 bytes $\sim$ 280 gigabytes. Note the size of the table.

The decision table for Therapy5 is even worse. It considers up to 5,309,410,000,000 different combinations! More

than 150 terabytes are needed in the file system. In short, it is a tough job to summarise the optimal policy content.

The partial evaluation of the whole diagram by instantiating the evidence on seventeen nodes alleviated somewhat the burden. The diagram may require up to 1,000 decision tables which needs an hour to evaluate each one, obtaining five final decision tables with size 18 megabytes, see below. The results are later composed incrementaly on a $KBM2L$ structure, as we explained above. Thus, it is *potentially* possible to provide inference procedures on partial instantiated evaluations of very complex models. We have shown how to do this by means of a learning mechanism.

V. Results

When dividing the evaluation process by instantiation of evidence, we choose the right number of attributes to get a manageable size for the decision tables and to produce the smallest impact on the propagation of uncertainty [4]. Remember in Section 4 that we had selected seventeen attributes ($ChildAge$, $ChildRhF$, $MotherRhF$, $ChildAB0F$, $MotherAB0F$, ...) for *IctNeo*, each with two, three or more values in their domains, i.e., several millions of instances. These instances yielded one thousand decision tables to add to a single $KBM2L$ list. The evaluation of each subproblem (problem instance) generated five tables with 9 (Therapy 1), 189 (Therapy 2), 3,969 (Therapy 3), 83,349 (Therapy 4) and 250,047 (Therapy 5) values stored in 18 megabytes. The base for the initial $KBM2L$ list, corresponding to Therapy 2 decision table after the learning procedure of the decision tables supplied by the instantiation, is shown below:

$$B_1 = \{CBrb1,\ CBrb2,\ Therapy1,\ CHgb,\ ChildAge,\ ChildRhF,\ MotherRhF,\ ChildAB0F,\ MotherAB0F,\ ApgarT,\ CRhFICT,\ MRhFICT,\ CAB0FICT,\ GestAge,\ BirthWeight,\ Prim,\ Instr,\ J,\ MotherAge,\ MotherRace,\ MotherDisease\}.$$

The learning process allows us to build the $KBM2L$ list for this base with the problem instances. The instances (1,000) are copied step by step over the $KBM2L$ list. Therapy 2 $KBM2L$ has 260 grains of knowledge covering the whole set of 189,000 evaluated cases included in the set of 4,013,162,496 combinations, the representation space of the decision table. We find $unKB$ items in the $KBM2L$ because we did not evaluate the whole problem. We evaluate an instance set which represents only the 0.0047 % of the table. The $unKB$ items are not interesting for the doctors and if some of them were interesting, then we would evaluate and learn the corresponding subproblems.

We use a two-column table to present a fragment of this list (see Table V). The left-hand column includes the proposal of the system (as the influence diagram has five decisions, there is a proposal for each one). The right-hand column shows the attributes associated with each proposal. The fixed part (the attributes without changes for their cases) of each item is highlighted in bold type.

The optimisation phase shows that the initial base is not optimal and that by changing the order of the attributes (the base) we could get a shorter list and refine the knowledge about the decisive attributes. After around 300 changes of base, we get a representation of the knowledge with only sixteen grains (items). The optimal base is:

$$B_f = \{CBrb1,\ CBrb2,\ Therapy1,\ CHgb,\ ChildAge,\ GestAge,\ BirthWeight,\ Prim,\ Instr,\ ChildRhF,\ MotherRhF,\ ChildAB0F,\ MotherAB0F,\ ApgarT,\ CRhFICT,\ MRhFICT,\ CAB0FICT,\ J,\ MotherAge,\ MotherRace,\ MotherDisease\}.$$

B_f leads to a shorter list, see Table VI. This list can be read as sixteen rules indicating the optimal global policy as a function of the key attributes, the fixed part of the item.

The results have achieved the following objectives:

1. A sizeable reduction in the memory space required to store the KB.
2. The reduction of the decision table by several orders of magnitude.
3. Attainment of explicit decision-making rules.
4. A means of observing the influence of parameter variation on the decision-making rules and, therefore, a practical procedure for performing sensitivity analysis.

VI. Discussion

In a decision-making problem, the preferences and the structure of dependencies and independencies represented in a decision model produce patterns of regularity in the proposals of the DSS. The granularity of the decision tables not only depends on the problem, but also on the internal organisation of the tables. As explained, a reorganisation may lead to a grouping of cases in contexts of identical proposals. A good organisation reduces the memory required to store the decision tables and brings out qualitative information about the variables: only some variables are really relevant in the contexts.

An interesting application of this technique is the analysis of decision models. We can consider some parameters (probabilities, preferences, dependencies, etc.) as attributes and evaluate the resultant models using different values. The data from the evaluation can be organised as $KBM2L$ lists and the relationships among the values of the parameters and the behaviour of the DSS can be analysed. This is a kind of sensitivity analysis.

The scope of our methodology is quite wide since it comprises learning, explanation, sampling of the representation space, inference from the partial knowledge obtained by means of instances, knowledge base construction, distributed DSSs, sensitivity analysis, etc.

The procedure presented in this paper is also applicable to situations where the data come from mathematical programs, data bases or data acquisition systems. Finally, we are working on many open issues: rules to manage the items of the list, operations among lists with different bases, techniques to find the optimal base, connections with multivariate analysis, etc.

Acknowledgments

Research supported by Project CAM 07T/0027/2000.

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, *Computers & Operations Research* 27, 7-8, 725-740, 2000.

[2] Bielza, C., Ríos-Insua, S., Gómez, M., Fdez del Pozo, J.A. Sensitivity Analysis in *IctNeo*, in *Robust Bayesian Analysis*, D. Ríos Insua, F. Ruggeri (eds.), Lecture Notes in Statistics, 152, 317-334, Springer, Berlin, 2000.

[3] Cooper, G.F. The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks, *Artificial Intelligence* 42, 393-405, 1990.

[4] Ezawa, K. Evidence Propagation and Value of Evidence On Influence Diagrams, *Operations Research* 46, 1, 73-83, 1998.

[5] Gómez, M., Ríos-Insua, S., Bielza, C., Fernández del Pozo, J.A. Multiattribute Utility Analysis in the *IctNeo* System, in *Research and Practice in Multiple Criteria Decision Making*, Y.Y. Haimes, R.E. Steuer (eds.), Lecture Notes in Economics and Mathematical Systems 487, 81-92, Springer, Berlin, 2000.

[6] Henrion, M., Breese, J.S., Horvitz, E.J. Decision Analysis and Expert Systems, *Artificial Intelligence Magazine* 12, 4, 64-91, 1991.

[7] Keeney, R. Decision Analysis: An Overview, *Operations Research* 30, 803-838, 1982.

[8] Kirkpatrick, S., Gelett, C.D., Vecchi, M.P. Optimization by Simulated Annealing, *Science* 220, 621-630, 1983.

[9] Pearl, J. (1988) *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Mateo, CA, 1988.

[10] Raiffa, H. (1968) *Decision Analysis: Introductory Lectures on Choices Under Uncertainty*, Addison-Wesley, Reading, MA, 1968.

[11] Ríos-Insua, S., Gómez. M., Bielza, C., Fdez del Pozo, J.A. Implementation of *IctNeo*: a Decision Support System for Jaundice Management, in *Operations Research Proceedings 1999*, K. Inderfurth, G. Schwödiauer, W. Domschke, F. Juhnke, P. Kleinschmidt, G. Wäscher (eds.), Gesellschaft für Operations Research e.V. (GOR), 554-559, Springer, Berlin, 2000.

[12] Shachter, R.D. (1986) Evaluating influence diagrams, *Operations Research* 34, 6, 871-882, 1986.

TABLE III
STRUCTURE OF THE TABLE FOR THE SECOND DECISION

Name of attributes	Domain of attributes
Therapy 2	NotTherapy2, Discharge, Observation12h Phototherapy6h, Phot12h, Phot24h BloodExchange
Instantiated attributes	
Child's Age (ChildAge)	less than 36 hours, between 36 and 72 hours, more than 72 hours
Child's Rh Factor (ChildRhF)	Negative, Positive
Mother's Rh Factor (MotherRhF)	Negative, Positive
Child's AB0 Group Factor (ChildAB0F)	ZERO, A, B, AB
Mother's AB0 Group Factor (MotherAB0F)	ZERO, A, B, AB
Apgar Test (ApgarT)	between 0 and 3, between 4 and 7, between 8 and 10
Child's Rh Factor Isoimmunization Coombs' Test (CRhFICT)	Negative, Positive
Mother's Rh Factor Isoimmunization Coombs' Test (MRhFICT)	Negative, Positive
Child's AB0 Group Isoimmunization Coombs' Test (CAB0FICT)	Negative, Positive
Gestational Age (GestAge)	less than 36 weeks, 36 weeks, more than 36 weeks
Birth Weight (BirthWeight)	between 500 and 1000 g, between 100 and 1500 g, between 1500 and 2500 g, more than 2500 g
Primiparous? (Prim)	Yes, No
Delivery with Instruments (Instr)	Natural, Instrumental
Jaundice (J)	Normal, Yellow, Yellow-Feet, Pumpkin-Yellow
Mother's Race (MotherRace)	Caucasian, Gipsy, Asian, Black
Mother's Age (MotherAge)	between 15 and 18 years, between 19 and 35 years, more than 35 years
Mother's Disease (MotherDisease)	no, yes
Non-Instantiated attributes	
Concentr. of Bilirubin 1 (CBrb1)	Normal, Pathological, Very Pathological
Concentr. of Bilirubin 2 (CBrb2)	Normal, Pathological, Very Pathological
Therapy 1	NotAdmission, Observation6h, Obs12h, Obs24h Phototherapy6h, Phot12h, Phot24h
Concentr. of Hemoglobin (CHgb)	Normal, Pathological, Very Pathological

TABLE IV
IctNeo PROTOCOL

Therapy 1		Therapy 2		Therapy 3	
NotAdmission	$U_{1,0}$	NotTherapy2	$U_{2,0}$	NotTherapy3	$U_{3,0}$
Observation6h	$U_{1,1}$	Discharge	$U_{2,1}$	Discharge	$U_{3,1}$
Observation12h	$U_{1,2}$	Observation12h	$U_{2,2}$	Observation12h	$U_{3,2}$
Observation24h	$U_{1,3}$	Phototherapy6h	$U_{2,3}$	Phototherapy6h	$U_{3,3}$
Phototherapy6	$U_{1,4}$	Phototherapy12h	$U_{2,4}$	Phototherapy12h	$U_{3,4}$
Phototherapy12h	$U_{1,5}$	Phototherapy24h	$U_{2,5}$	Phototherapy24h	$U_{3,5}$
Phototherapy24h	$U_{1,6}$	BloodExchange	$U_{2,6}$	BloodExchange	$U_{3,6}$

Therapy 4		Therapy 5	
NotTherapy4	$U_{4,0}$	NotTherapy5	$U_{5,0}$
Discharge	$U_{4,1}$	Discharge	$U_{5,1}$
Observation12h	$U_{4,2}$	Continued	$U_{5,2}$
Phototherapy12h	$U_{4,3}$	——-	——-
Phototherapy24h	$U_{4,4}$	——-	——-

TABLE V
KBM2L LIST OF THERAPY2 DECISION TABLE, 260 GRAINS OF KNOWLEDGE

unKB	...
Therapy2: NotTherapy2 Item-1	**CBrb2:Normal,CBrb1:Normal,Therapy1:NotAdmission, CHgb:Normal,ChilAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Natural,** ChildRhF:Negative,MotherRhF:Negative, ChildAB0F:AB,MotherABOF:AB,ApgarT5:between 8 and 10, CRhFICT:Negative,MRhFICT:Negative,CAB0FICT:Negative, J:Yellow,MotherAge:between 19 and 35,MotherRace:Caucasian, MotherDisease:no
unKB	...
Therapy2: Observation6h Item-2	**CBrb2:Normal,CBrb1:Pathological,Therapy1:Phototherapy6, CHgb:Normal,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Natural,** ...
unKB	...
Therapy2: NotTherapy2 Item-3	**CBrb2:Normal,CBrb1:Normal,Therapy1:NotAdmission, CHgb:Normal,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Natural,** ...
unKB	...
Therapy2: Observation6h Item-4	**CBrb2:Normal,CBrb1:Pathological,Therapy1:Phototherapy6, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Natural,** ...
unKB	...
Therapy2: NotTherapy2 Item-5	**CBrb2:Normal,CBrb1:Normal,Therapy1:NotAdmission, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Natural,** ...
unKB	...
Therapy2: Discharge Item-6	**CBrb2:Normal,CBrb1:High,Therapy1:Phototherapy6, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:Yes,Instr:Natural,** ...
unKB	...
Therapy2: NotTherapy2 Item-7	**CBrb2:Pathological,CBrb1:Normal,Therapy1:NotAdmission, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Natural,** ...
unKB	...
Therapy2: BloodExchange Item-8	**CBrb2:Very Pathological,CBrb1:Very Pathological,Therapy1:Phototherapy6, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:less than 36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Natural,** ...
unKB	...
Therapy2: NotTherapy2 Item-9	**CBrb2:Normal,CBrb1:Normal,Therapy1:NotAdmission, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Instrumental,** ...
unKB	...
Therapy2: BloodExchange Item-10	**CBrb2:Very Pathological,CBrb1:Very Pathological,Therapy1:Phototherapy6, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:Yes,Instr:Instrumental,** ...
unKB	...
Therapy2: Phototherapy12h Item-11	**CBrb2:Pathological,CBrb1:Very Pathological,Therapy1:NotAdmission, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:No,Instr:Instrumental,** ...
unKB	...
Therapy2: BloodExchange Item-12	**CBrb2:Very Pathological,CBrb1:Very Pathological,Therapy1:Phototherapy6, CHgb:Pathological,ChildAge1:less than 36 hours,GestAge:36 weeks, BirthWeight:more than 2500 g,Prim:Yes,Instr:Instrumental,** ...
...	...

TABLE VI

IctNeo SYSTEM PROTOCOL (THERAPY2), 16 GRAINS OF KNOWLEDGE

item	Evaluated Attributes	Optimal Therapy2
1	CBrb2=Normal, CBrb1=Normal, CHgb=Normal, Therapy1=NotAdmission, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	NotTherapy2
	...	$unKB$
2	CBrb2=Normal, CBrb1=Normal, CHgb=Normal, Therapy1=Observation6h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Discharge
	...	$unKB$
3	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Observation12h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Discharge
4	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Phototherapy6h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Observation6h
5	CBrb2=Normal, CBrb1=Pathological, CHgb=Pathological, Therapy1=Phototherapy6h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Observation12h
6	CBrb2=Normal, CBrb1=Pathological, CHgb=Pathological, Therapy1=Phototherapy12h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Phototherapy24h
7	CBrb2=Pathological, CBrb1=Pathological, CHgb=Pathological, Therapy1=Phototherapy24h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Phototherapy24h
	...	$unKB$
8	CBrb2=Pathological, CBrb1=Pathological, CHgb=Pathological, Therapy1=Phototherapy12h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Observation12h
9	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Phototherapy24h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Phototherapy24h
	...	$unKB$
10	CBrb2=Normal, CBrb1=Pathological, CHgb=Pathological, Therapy1=Phototherapy12h, GestAge:less than 36 weeks, BirthWeight:more than 2500 g, Prim:No	Phototherapy24h
11	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Phototherapy24h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Phototherapy12h
	...	$unKB$
12	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Phototherapy12h, GestAge:more than 36 weeks, BirthWeight:more than 2500 g, Prim:No	Observation12h
13	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Phototherapy6h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	Phototherapy12h
14	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Observation12h, GestAge:more than 36 weeks, BirthWeight:more than 2500 g, Prim:No	Discharge
15	CBrb2=Normal, CBrb1=Pathological, CHgb=Normal, Therapy1=Phototherapy24h, GestAge:less than 36 weeks, BirthWeight:more than 2500 g, Prim:No	Observation12h
	...	$unKB$
16	CBrb2=Very Pathological, CBrb1=Pathological, CHgb=Very Pathological, Therapy1=Phototherapy24h, GestAge:36 weeks, BirthWeight:more than 2500 g, Prim:No	BloodExchange
	...	$unKB$