* For the reference see the last page.

at the heart of *MINOS* (see [8], [9]) one of the most widely used methods for solving problems with nonlinear constraints. Other transformation methods transform the problem to that of solving a sequence of unconstrained or bounds-constrained problem. Transformation methods have an advantage of over direct methods when developing software. For example, if you have a method for solving large-scale linearly-constrained problems then it can be used as a kernel in an algorithm to solve large-scale nonlinearly-constrained problems.

See also: **Equality-constrained nonlinear programming: KKT necessary optimality conditions; Second order optimality conditions for nonlinear optimization; Lagrangian duality: Basics; Saddle point theory and optimality conditions; First order constraint qualifications; Second order constraint qualifications; Kuhn–Tucker optimality conditions; Rosen's method, global convergence, and Powell's conjecture; History of optimization; Redundancy in nonlinear programs; Relaxation in projection methods; SSC minimization algorithms; SSC minimization algorithms for nonsmooth and stochastic optimization.**

References

[1] BAZARAA, M.S., SHERALI, H.D., AND SHETTY, C.M.: *Nonlinear programming: Theory and algorithms*, second ed., Wiley, 1993.

[2] BERTSEKAS, D.P.: *Nonlinear programming*, second ed., Athena Sci., 1999.

[3] FLETCHER, R.: *Practical methods of optimization*, Wiley, 1988.

[4] GILL, P.E., MURRAY, W., AND WRIGHT, M.H.: *Practical optimization*, Acad. Press, 1981.

[5] KARUSH, W.: 'Minima of functions of several variables with inequalities as side constraints', *Dept. Math. Univ. Chicago* (1939).

[6] KUHN, H.W.: 'Nonlinear programming: A historical note', in J.K. LENSTRA, A.H.G. RINNOOY KAN, AND A. SCHRIJVER (eds.): *History of Mathematical Programming: A Collection of Personal Reminiscences*, Elsevier, 1991, pp. 82–96.

[7] KUHN, H.W., AND TUCKER, A.W.: 'Nonlinear programming', in J. NEYMAN (ed.): *Proc. Second Berkeley Symposium on Mathematical Statistics and Probability*, Univ. Calif. Press, 1951, pp. 481–492.

[8] MURTAGH, B.A., AND SAUNDERS, M.A.: 'A projected Lagrangian algorithm and its implementation for sparse nonlinear constraints', *Math. Program. Stud.* **16** (1982), 84–117.

[9] MURTAGH, B.A., AND SAUNDERS, M.A.: 'MINOS 5.4 user's guide', *SOL Report Dept. Oper. Res. Stanford Univ.* **83-20R** (1995).

[10] NASH, S.G., AND SOFER, A.: *Linear and nonlinear programming*, McGraw–Hill, 1996.

Walter Murray
Stanford Univ.
Stanford, USA
E-mail address: walter@stanford.edu

MSC2000: 49M37, 65K05, 90C30
Key words and phrases: constrained optimization, optimality conditions, algorithms.

INFERENCE OF MONOTONE BOOLEAN FUNCTIONS by Triantaphyllou

The goal in a classification problem is to uncover a system that places examples into two or more mutually exclusive groups. Identifying a classification system is beneficial in several ways. First of all, examples can be organized in a meaningful way, which will make the exploration and retrieval of examples belonging to specific group(s) more efficient. The tree-like directory structure, used by personal computers in organizing files, is an example of a classification system which enables users to locate files quickly by traversing the directory paths. A classification system can make the relations between the examples easy to understand and interpret. A poor classification strategy, on the other hand, may propose arbitrary, confusing meaningless relations. An extracted classification system can be used to classify new examples. an incomplete or stochastic system, its struc- may pose questions whose answers may gene- the system or make it more accurate.

A special type of *classification problem*, the *Boolean function inference problem*, is all the examples are represented by binary 1) attributes and each example belongs to two categories. Many other types of classification problems may be converted into a Boolean tion inference problem. For example, a multi- gory classification problem may be converted several two-category problems. In a similar ion, example attributes can be converted set of binary variables.

solving the Boolean function inference problem, many properties of Boolean logic are directly applicable. A *Boolean function* will assign a binary value to each Boolean vector (example). See for an overview of Boolean functions. Usually, a Boolean function is expressed as a conjunction of disjunctions, called the *conjunctive normal form* (CNF), or a disjunction of conjunctions, called the *disjunctive normal form* (DNF). CNF can be written as:

$$\bigwedge_{j=1}^{k} \left(\bigvee_{i \in \rho_j} x_i \right),$$

where x_i is either the attribute or its negation, k is the number of attribute disjunctions and ρ_j is the jth index set for the jth attribute disjunction. Similarly, DNF can be written as:

$$\bigvee_{j=1}^{k} \left(\bigwedge_{i \in \rho_j} x_i \right).$$

It is well known that any Boolean function can be written in CNF or DNF form. See [20] for an algorithm converting any Boolean expression into CNF. Two functions in different forms are regarded as equivalent as long as they assign the same function values to all the Boolean vectors. However, placing every example into the correct category is only one part of the task. The other part is to make the classification criteria meaningful and understandable. That is, an inferred Boolean function should be as simple as possible. One part of the Boolean function inference problem that has received substantial research efforts is that of simplifying the representation of Boolean functions, while maintaining a general representation power.

Inference of Monotone Boolean Functions. When the target function can be any Boolean function with n attributes, all of the 2^n examples have to be examined to reconstruct the entire function. When we have a priori knowledge about the subclass of Boolean functions the target function belongs to, on the other hand, it may be possible to reconstruct it using a subset of the examples. Often one can obtain the function values on examples one by one. That is, at each inference step, an example is posed as a question to an oracle, which, in return, provides the correct function value. A function, f, can be defined by its oracle A_f which, when fed with a vector $x = \langle x_1, \ldots, x_n \rangle$, returns its value $f(x)$. The inference of a Boolean function from questions and answers is known as *interactive learning of Boolean functions*. In many cases, especially when it is either difficult or costly to query the oracle, it is desirable to pose as few questions as possible. Therefore, the choice of examples should be based on the previously classified examples.

The *monotone Boolean functions* form a subset of the Boolean functions that have been extensively studied not only because of their wide range of applications (see [2], [7], [8] and [24]) but also their intuitive interpretation. Each attribute's contribution to a monotone function is either nonnegative or nonpositive (not both). Furthermore, if all of the attributes have nonnegative (or nonpositive) effects on the function value then the underlying monotone Boolean function is referred to as *isotone* (respectively *antitone*). Any isotone function can be expressed in DNF without using negated attributes. In combinatorial mathematics, the set of isotone Boolean functions is often represented by the *free distributive lattice* (FDL). To formally define monotone Boolean function, consider ordering the binary vectors as follows [21]:

Definition 1 Let E^n denote the set of all binary vectors of length n; let x and y be two such vectors. Then, the vector $x = \langle x_1, \ldots, x_n \rangle$ *precedes* vector $y = \langle y_1, \ldots, y_n \rangle$ (denoted as $x \preceq y$) if and only if $x_i \leq y_i$ for $1 \leq i \leq n$. If, at the same time $x \neq y$, then x *strictly precedes* y (denoted as $x \prec y$). □

According to this definition, the order of vectors in E^2 can be listed as follows:

$$\langle 11 \rangle \prec \langle 01 \rangle \prec \langle 00 \rangle$$

and

$$\langle 11 \rangle \prec \langle 10 \rangle \prec \langle 00 \rangle .$$

Note that the vectors $\langle 01 \rangle$ and $\langle 10 \rangle$ are in a sense incomparable.

Based on the order of the Boolean vectors, a nondecreasing monotone (isotone) Boolean function can be defined as follows [21]:

DEFINITION 2 A Boolean function f is said to be an *nondecreasing monotone Boolean function* if and only if for any vectors $x, y \in E^n$, such that $x \prec y$, then $f(x) \prec f(y)$. □

A *nonincreasing monotone* (antitone) Boolean function can be defined in a similar fashion. As the method used to infer an antitone Boolean function is the same as that of a isotone Boolean function, we will restrict our attention to the isotone Boolean functions.

When analyzing a subclass of Boolean functions, it is always informative to determine its size. This may give some indications of how general the functions are and how hard it is to infer them. The number of isotone Boolean functions, $\Psi(n)$, defined on E^n is sometimes referred to as the nth *Dedekind number* after R. Dedekind, [6] who computed it for $n = 4$. Since then it has been computed for up to E^8.

- $\Psi(1) = 3$;
- $\Psi(2) = 6$;
- $\Psi(3) = 20$;
- $\Psi(4) = 168$ [6];
- $\Psi(5) = 7,581$ [4];
- $\Psi(6) = 7,828,354$ [28];
- $\Psi(7) = 2,414,682,040,998$ [5];
- $\Psi(8) = 56,130,437,228,687,557,907,788$ [29].

Wiedeman's algorithm [29] employed a Cray-2 processor for 200 hours to compute the value for $n = 8$. This gives a flavor of the complexity of computing the exact number of isotone Boolean functions. The computational infeasibility for larger values of n provides the motivation for approximations and bounds. The best known bound on $\Psi(n)$ is due to D. Kleitman, [12] and Kleitman and G. Markowsky, [13]:

$$\Psi(n) \leq 2^{\binom{n}{\lfloor n/2 \rfloor}\left(1 + c\frac{\log n}{n}\right)},$$

where c is a constant and $\lfloor n/2 \rfloor$ is the integer part of $n/2$.

This bound, which is an improvement over the first bound obtained by G. Hansel, [11], are also based on the Hansel chains described below. Even though these bounds can lead to good approximations for $\Psi(n)$, when n is large, the best known asymptotic is due to A.D. Korshunov, [15]:

$$\Psi(n) \sim \begin{cases} 2^{\binom{n}{n/2}} e^{f(n)} & \text{for even } n, \\ 2^{\binom{n}{n/2 - 1/2} + 1} e^{g(n)} & \text{for odd } n, \end{cases}$$

where

$$f(n) = \binom{n}{n/2 - 1}\left(\frac{1}{2^{n/2}} + \frac{n^2}{2^{n+5}} - \frac{n}{2^{n+4}}\right),$$

$$g(n)$$

$$= \binom{n}{n/2 - 3/2}\left(\frac{1}{2^{(n+3)/2}} - \frac{n^2}{2^{n+6}} - \frac{n}{2^n + 3}\right) + \binom{n}{n/2 - 1/2}\left(\frac{1}{2^{(n+1)/2}} + \frac{n^2}{2^{n+4}}\right).$$

I. Shmulevich [24] achieved a similar but slightly inferior asymptotic for even n in a simpler and more elegant manner, which led to some interesting distributional conjectures regarding isotone Boolean functions.

Even though the number of isotone Boolean functions is large, it is a small fraction of the number of general Boolean functions, 2^{2^n}. This is the first hint towards the feasibility of efficiently inferring monotone Boolean functions. Intuitively, one would conjecture that the generality of this class was sacrificed. That is true, however, a general Boolean function consists of a set of areas where it is monotone. In fact, any Boolean function $q(x_1, \ldots, x_n)$ can be represented by several nondecreasing $g_i(x)$ and nonincreasing $h_j(x)$ monotone Boolean functions in the following manner [17]:

$$q(x) = \bigvee_i \left(g_i(x) \bigwedge_j h_j(x) \right).$$

As a result, one may be able to solve the general Boolean function inference problem by considering several *monotone Boolean function inference* problems. Intuitively, the closer the target function is to a monotone Boolean function, the fewer monotone Boolean functions are needed to represent it and more successful this approach might be. In [17] the problem of joint restoration of [illegible] nested monotone Boolean functions f_1 and f_2 [illegible] stated. The approach in [17] allows one to further decrease the dialogue with an expert (*oracle*) [illegible] restore a complex function of the form $f_1\&$[illegible] which is not necessarily monotone.

The Shannon Function and the Hansel Theorem. The complexity of inferring isotone Boolean functions was mentioned in the previous section, when realizing that the number of isotone Boolean functions is a small fraction of the total number of general Boolean functions. In defining the most common complexity measure for the Boolean function inference problem, consider the following notation. Let M_n denote the set of all monotone Boolean functions, and $A = \{F\}$ be the set of all algorithms which infer $f \in M_n$, and $\varphi(F, f)$ be the number of questions to the oracle A_f required to infer f. The *Shannon function* $\varphi(n)$ can be introduced as follows [14]:

$$\varphi(n) = \min_{F \in A} \left(\max_{f \in M_n} \varphi(F, f) \right).$$

An upper bound on the number of questions needed to restore a monotone Boolean function is given by the following equation (known as the *Hansel theorem*) [11]:

$$\varphi(n) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}.$$

That is, if a proper *question-asking strategy* is applied, the total number of questions needed to infer any monotone Boolean function should not exceed $\varphi(n)$. The Hansel theorem can be viewed as the worst-case scenario analysis. Recall, from the previous section, that all of the 2^n questions are necessary to restore a general Boolean function. D.N. Gainanov [9] proposed three other criteria for evaluating the efficiency of algorithms used to infer isotone Boolean functions. One of them is the average case scenario and the two others consider two different ways of normalizing the Shannon function by the size of the target function.

Hansel Chains. The vectors in E^n can be placed in chains (sequences of vectors) according to *monotonicity*. The Hansel chains is a particular set of chains that can be formed using a dimensionally recursive algorithm [11]. It starts with the single Hansel chain in E^1:

$$H^1 = \{\langle 0 \rangle, \langle 1 \rangle\}.$$

To form the Hansel chains in E^2, three steps are required, as follows:

1 Attach the element '0' to the front of each vector in H^1 and get chain $C^{2\,\min} = \{\langle 00 \rangle, \langle 01 \rangle\}$.
2 Attach the element '1' to the front of each vector in H^1 and get chain $C^{2\,\max} = \{\langle 10 \rangle, \langle 11 \rangle\}$.
3 Move the last vector in $C^{2\,\max}$, i.e. vector $\langle 11 \rangle$, to the end of $C^{2\,\min}$: $H^{2,1} = \{\langle 00 \rangle, \langle 01 \rangle, \langle 11 \rangle\}$; $H^{2,2} = \{\langle 10 \rangle\}$.

To form the Hansel chains in E^3, these steps are repeated:

1 $C^{3,1\,\min} = \{\langle 000 \rangle, \langle 001 \rangle, \langle 011 \rangle\}$;
$C^{3,2\,\min} = \{\langle 010 \rangle\}$.
2 $C^{3,1\,\max} = \{\langle 100 \rangle, \langle 101 \rangle, \langle 111 \rangle\}$;
$C^{3,2\,\max} = \{\langle 110 \rangle\}$.
3 $H^{3,1} = \{\langle 000 \rangle, \langle 001 \rangle, \langle 011 \rangle, \langle 111 \rangle\}$;
$H^{3,2} = \{\langle 100 \rangle, \langle 101 \rangle\}$;
$H^{3,3} = \{\langle 010 \rangle, \langle 110 \rangle\}$.

Note that since there is only one vector in the $C^{3,2\,\max}$ chain, it can be deleted after the vector $\langle 110 \rangle$ is moved to $C^{3,2\,\min}$. This leaves the three chains listed in Table 1. In general, the Hansel chains in E^n can be generated recursively from the Hansel chains in E^{n-1} by following the three steps described above.

chain #	vector in-chain index	vector
1	1	000
	2	001
	3	011
	4	111
2	1	100
	2	101
3	1	010
	2	110

Table 1: Hansel chains for E^3.

A nice property of the Hansel chains is that all the vectors in a particular chain are arranged in increasing order. That is, if the vectors V_j and V_k are in the same chain then $V_j < V_k$ (i.e., V_j strictly precedes V_k when $j < k$). Therefore, if the underlying Boolean function is isotone, then one can classify vectors within a chain easily. For example, if a vector V_j is negative (i.e., $f(V_j) = 0$), then all the vectors preceding V_j in the same chain are also negative (i.e., $f(V_k) = 0$ for any $k < j$). Similarly, if a vector V_j is positive, then all the vectors succeeding V_j in the same chain are also positive. The monotone ordering of the vectors in Hansel chains motivates the composition of an efficient question-asking strategy discussed in the next section.

Devising a Smart Question-Asking Strategy. The most straightforward question-asking strategy, which uses Hansel chains, sequentially moves from chain to chain. Within each chain one may also sequentially select vectors to pose as questions. After an answer is given, the vectors (in other chains also) that are classified as a result of monotonicity are eliminated from further questioning. Once all the vectors have been eliminated, the underlying function is revealed. The maximum number of questions for this method, called the *sequential Hansel chains question-asking strategy*, will not exceed the upper limit $\varphi(n)$, given in the Hansel theorem, as long as the chains are searched in increasing size.

Although the sequential question-asking strategy is easy to implement and effective in reducing the total number of questions, there is still room for improvements. N.A. Sokolov [25] introduced an algorithm that sequentially moves between the Hansel chains in decreasing size and performs a middle vector search of each chain. His algorithm does not require storing all the Hansel chains since at each iteration it only requires a single chain. This advantage is obtained at the cost of asking more questions than needed.

In an entirely different approach, Gainanov [9] presented a heuristic that has been used in numerous algorithms for inferring a monotone Boolean function, such as in [3] and in [18]. This heuristic takes as input an unclassified vector and finds a border vector (maximal false or minimal true) by sequentially questioning neighboring vectors. The problem with most of the inference algorithms based on this heuristic is that they do not keep track of the vectors classified, only the resulting border vectors. Note that for an execution of this heuristic, all of the vectors questioned are not necessarily covered by the resulting border vector, implying that valuable information may be lost. In fact, several border vectors may be unveiled during a single execution of this heuristic, but only one is stored. Many of these methods are designed to solve large problems where it might be inefficient or even infeasible to store all of the information gained within the execution of the heuristic. However, these methods are not efficient (not even for small size problems), in terms of the number of queries they require.

One may look at each vector as carrying a 'reward' value in terms of the number of other vectors that will be classified concurrently. This reward value is a random variable that takes on one of two (one if the two values are the same) values depending on whether the vector is a positive or a negative example of the target function. The expected reward is somewhere between these two possible values. If one wishes to maximize the expected number of classified vectors at each step, the probabilities associated with each of these two values need to be computed in addition to the actual values. Finding the exact probabilities is hard, while finding the reward values is relatively simple for a small set of examples.

This is one of the underlying ideas for the new inference algorithm termed the *binary search-Hansel chains question-asking strategy*. This method draws its motivation, for calculating and comparing the 'reward' values for the middle vectors in each Hansel chain, from the widely used *binary search algorithm* (see, for instance, [19]). Within a given chain, a binary search will dramatically reduce the number of questions (to the order of $\log_2$ while the sequential search is linear). Once the 'reward' values of all the middle vectors have been found, the most promising one will be posed as a question to the oracle. Because each vector has two values, selecting the most promising vector is subjective and several different evaluative criteria can be used.

The binary search-Hansel chains question-asking strategy can be divided into the following steps:

1) Select the middle vector of the unclassified vectors in each Hansel chain.

2) Calculate the reward values for each middle vector. That is, calculate the number of vectors that can be classified as positive (denoted as P) if it is positive and negative (denoted as N) if it is negative.

3) Select the most promising middle vector based on the (P,N) pairs of the middle vectors, and ask the oracle for its membership value.

4) Based on the answer in Step 3, eliminate all the vectors that can be classified as a result of the previous answer and the property of monotonicity.

5) Redefine the middle vectors in each chain as necessary.

6) Unless all the vectors have been classified, go back to Step 2.

The inference of a monotone Boolean function on E^3 by using the binary search-Hansel chains question-asking strategy is illustrated below. The specifics of Iteration 1, described below, are also shown in Table 2. At the beginning of first iteration, the middle vectors in each Hansel chain (as described in Step 1) are selected and marked with the '←' symbol in Table 2. Then, according to Step 2, the reward value for each one of these middle vectors is calculated. For instance, if ⟨001⟩ (the second vector in chain 1) has a function value of 1, then the three vectors ⟨000⟩, ⟨001⟩ and ⟨010⟩ are also classified as positive. That is, the value of P for vector ⟨001⟩ equals 4. Similarly, ⟨000⟩ will be classified as 0 if ⟨001⟩ is classified as 0 and thus its reward value N equals 2.

Once the 'reward' values of all the middle vectors have been evaluated, the most promising middle vector will be selected based on their (P, N) pairs. Here we choose the vector whose $\min(P, N)$ value is the largest among the middle vectors. If there is a tie, it will be broken randomly. Based on this evaluative criterion, vector 2 is chosen in chain 1 and is marked with '←' in the column 'selected middle vector with the largest $\min(P, N)$'. After receiving the function value of 1 for vector ⟨001⟩, its value is placed in the 'answer' column. This answer is used to eliminate all of the vectors succeeding ⟨001⟩. The middle vector in the remaining chains are updated as needed. At least one more iteration is required, as there still are unclassified vectors.

After the second iteration, no unclassified vectors are left in chains 1 and 2, and the middle of these chains need not be considered anymore. Therefore, an 'X' is placed in the column called 'middle vector in the chain' in Table 4. At the beginning of the third iteration, the vector ⟨010⟩ is chosen and the function value of the remaining two vectors ⟨010⟩ and ⟨110⟩ are determined. At this point all the vectors have been classified and the question-asking process stops.

The algorithm posed a total of three questions in order to classify all the examples. The final classifications listed in Table 5 corresponds to the monotone Boolean function $x_2 \bigvee x_3$.

Conclusions. This paper described some approaches and some of the latest developments in the problem of inferring monotone Boolean functions. As it has been described here, by using Hansel chains in the sequential question-asking strategy, the number of questions will not exceed the upper bound stated in the Hansel theorem. However, by combining the binary search of Hansel chains with the notion of an evaluative criterion, the number of questions asked can be further reduced. At present, the binary search-Hansel chains question-asking strategy is only applied to Hansel chains with a dimension of less than 10. However, it is expected that this method can be applied to infer monotone Boolean functions of larger dimensions with slight modifications.

See also: **Boolean and fuzzy relations; Checklist paradigm semantics for fuzzy logics; Alternative set theory; Finite complete systems of many-valued logic algebras; Optimization in Boolean classification problems; Optimization in classifying text documents.**

References

[1] ALEKSEEV, D.V.B.: 'Monotone Boolean functions': *Encycl. Math.*, Vol. 6, Kluwer Acad. Publ., 1988, pp. 306–307.

[2] BIOCH, J.C., AND IBARAKI, T.: 'Complexity of identifixation and dualization of positive Boolean functions', *Inform. and Comput.* **123** (1995), 50–63.

[3] BOROS, E., HAMMER, P.L., IBARAKI, T., AND KAWAKAMI, K.: 'Polynomial-time recognition of 2-monotonic positive Boolean functions given by an oracle', *SIAM J. Comput.* **26**, no. 1 (1997), 93–109.

[4] CHURCH, R.: 'Numerical analysis of certain free distributive structures', *J. Duke Math.* **9** (1940), 732–734.

[5] CHURCH, R.: 'Enumeration by rank of the elements of the free distributive lattice with 7 generators', *Notices Amer. Math. Soc.* **12** (1965), 724.

[6] DEDEKIND, R.: 'Ueber Zerlegungen von Zahlen durch ihre grössten gemeinsamen Teiler', *Festchrift Hoch. Braunschweig u. Ges. Werke, II* (1897), 103–148.

chain #	index of vectors in the chain	vector	vector classified	middle vector in the chain	reward P if the vector is positive	reward N if the vector is negative	selected middle vector with the largest $\min(P, N)$	answer	other vectors determined
1	1	000							
	2	001		←	4	2	←	1	
	3	011							1
	4	111							1
2	1	100		←	4	2			
	2	101							1
3	1	010		←	4	2			
	2	110							

Table 2: Iteration 1.

chain #	index of vectors in the chain	vector	vector classified	middle vector in the chain	reward P if the vector is positive	reward N if the vector is negative	selected middle vector with the largest $\min(P, N)$	answer	other vectors determined
1	1	000		←	4	1			0
	2	001	1						
	3	011	1						
	4	111	1						
2	1	100		←	2	2	←	0	
	2	101	1						
3	1	010		←	2	2			
	2	110							

Table 3: Iteration 2. The vector ⟨100⟩ is chosen and based on the answer, the class membership of the vectors ⟨100⟩ and ⟨000⟩ is determined.

chain #	index of vectors in the chain	vector	vector classified	middle vector in the chain	reward P if the vector is positive	reward N if the vector is negative	selected middle vector with the largest $\min(P, N)$	answer	other vectors determined
1	1	000	0						
	2	001	1	X					
	3	011	1						
	4	111	1						
2	1	100	0	X					
	2	101	1						
3	1	010		←	2	1	←	1	
	2	110							1

Table 4: Iteration 3.

chain #	vector in-chain index	vector	function value
1	1	100	0
	2	101	1
2	1	010	1
	2	110	1
3	1	000	0
	2	001	1
	3	011	1
	4	111	1

Table 5: The resulting class memberships.

[7] Eiter, T., and Gottlob, G.: 'Identifying the minimal transversals of a hybergraph and related problems', *SIAM J. Comput.* **24, number6** (1995), 1278–1304.

[8] Fredman, M.L., and Khachiyan, L.: 'On the complexity of dualization of monotone disjunctive normal forms', *J. Algorithms* **21** (1996), 618–628.

[9] Gainanov, D.N.: 'On one criterion of the optimality of an algorithm for evaluating monotonic Boolean functions', *USSR Comput. Math. Math. Phys.* **24**, no. 4 (1984), 176– 181.

[10] Gorbunov, Y., and Kovalerchuk, B.: 'An interactive method of monotone Boolean function restoration', *J. Acad. Sci. USSR Engin.* **2** (1982), 3–6, in Russian.

[11] Hansel, G.: 'Sur le nombre des fonctions Boolenes monotones den variables', *C.R. Acad. Sci. Paris* **262**, no. 20 (1966), 1088–1090.

[12] Kleitman, D.: 'On Dedekind's problem: The number of monotone Boolean functions', *Proc. Amer. Math. Soc.* **21**, no. 3 (1969), 677–682.

[13] Kleitman, D., and Markowsky, G.: 'On Dedekind's problem: The number of isotone Boolean functions. II', *Trans. Amer. Math. Soc.* **213** (1975), 373–390.

[14] Korobkov, V.K.: 'On monotone functions of the algebra of logic', *Probl. Kibernet.* **13** (1965), 5–28, in Russian.

[15] Korshunov, A.D.: 'On the number of monotone Boolean functions', *Probl. Kibernet.* **38** (1981), 5–108. (In Russian.)

[16] Kovalerchuk, B., Triantaphyllou, E., Deshpande, A.S., and Vityaev, E.: 'Interactive learning of monotone Boolean functions', *Inform. Sci.* **94**, no. 1-4 (1996), 87–118.

[17] Kovalerchuk, B., Triantaphyllou, E., and Vityaev, E.: 'Monotone Boolean functions learning techniques integrated with user interaction': *Proc. Workshop Learning from Examples vs. Programming by Demonstrations: Tahoe City, CA, USA*, 1995, pp. 41–48.

[18] Makino, K., and Ibaraki, T.: 'The maximum latency and identification of positive Boolean functions', *SIAM J. Comput.* **26**, no. 5 (1997), 1363–1383.

[19] Neapolitan, R., and Naimipour, K.: *Foundations of algorithms*, D.C. Heath, 1996.

[20] Peysakh, J.: 'A fast algorithm to convert Boolean expressions into CNF', *Techn. Report IBM Computer Sci. RC 12913*, no. 57971 (1987).

[21] Rudeanu, S.: *Boolean functions and equations*, North-Holland, 1974.

[22] Schneeweiss, W.G.: *Boolean functions: With engineering applications and computer applications*, Springer, 1989.

[23] Schneeweiss, W.G.: 'A necessary and sufficient criterion for the monotonicity of Boolean functions with deterministic and stochastic applications', *IEEE Trans. Computers* **45**, no. 11 (1996), 1300–1302.

[24] Shmulevich, I.: 'Properties and applications of monotone Boolean functions and stack filters', *PhD Thesis Dept. Electrical Engin. Purdue Univ.* (1997), http://shay.ecn.purdue.edu/~shmulevi/.

[25] Sokolov, N.A.: 'On the optimal evaluation of monotonic Boolean functions', *USSR Comput. Math. Math. Phys.* **22**, no. 2 (1982), 207–220.

[26] Torvik, V.I., and Triantaphyllou, E.: 'Minimizing the average query complexity of learning monotone Boolean functions', *Working Paper Dept. Industrial and Manuf. Syst. Engin., Louisiana State Univ.* (2000).

[27] Triantaphyllou, E., and Lu, J.: 'The knowledge acquisition problem in monotone Boolean systems', in A. Kent and J.G. Williams (eds.): *Encycl. Computer Sci. and Techn.*, M. Dekker, 1998.

[28] Ward, M.: 'Note on the order of free distributive lattices', *Bull. Amer. Math. Soc.* **52**, no. Abstract 135 (1946), 423.

[29] Wiedemann, D.: 'A computation of the eight Dedekind number', *Order* **8** (1991), 5–6.

Vetle I. Torvik
Dept. Industrial and Manufacturing Systems Engin.
3128 CEBA Building
Louisiana State Univ.
Baton Rouge, LA 70803–6409, USA
E-mail address: `vtorvik@unix1.sncc.lsu.edu`

Evangelos Triantaphyllou
Dept. Industrial and Manufacturing Systems Engin.
3128 CEBA Building
Louisiana State Univ.

Baton Rouge, LA 70803-6409, USA
E-mail address: `trianta@lsu.edu`
Web address: `www.imse.lsu.edu/vangelis`

MSC 2000: 90C09
Key words and phrases: Boolean function, monotone Boolean function, isotone Boolean function, antitone Boolean function, classification problem, Boolean function inference problem, free distributive lattice, conjunctive normal form, CNF, disjunctive normal form, DNF, interactive learning of Boolean functions, Shannon function, Hansel theorem, Hansel chain, sequential Hansel chains question-asking strategy, binary search-Hansel chains question-asking strategy, binary search.

INFINITE HORIZON CONTROL AND DYNAMIC GAMES, *IHDG*

In economics or biology there is no natural end time for a process. Nations as well as species have a very long future to consider. A mathematical abstraction for this phenomenon is the concept of *infinite time horizon* simply defined as an unbounded time interval of the form $[0, +\infty)$. The study of competing agents in a dynamic deterministic setting over a long time period can be cast in the framework of an infinite horizon dynamic game. This game is defined by the following 'objects':

- A system evolving over an infinite horizon is characterized by a *state* $x \in X \subseteq \mathbf{R}^{m_0}$. Some agents also called the *players* $i = 1, \ldots, p$ can influence the state's evolution through the choice of an appropriate *control* in an admissible class. The control value at a given time n for player i is denoted $u_i(n) \in U_i \subseteq \mathbf{R}^{m_i}$.
- The state evolution of such a dynamical system may be described either as a difference equation, if discrete time is used, or a differential equation in a continuous time framework. For definiteness we fix our attention here on a stationary difference equation and merely remark that similar comments apply for the case when other types of dynamical systems are considered.

  $$x(n+1) = f(x(n), u_1(n), \ldots, u_p(n))$$

  for $n = 0, 1, \ldots$, where $f\colon \mathbf{R}^{m_0} \times \ldots \times \mathbf{R}^{m_p} \to \mathbf{R}^{m_0}$ is a given state transition function.
- We assume that the agents can observe the state of the system and remember the *history* of the system evolution up to the current time n, that is, the sequence

  $$h_n = \{x(0), \mathbf{u}(0), \ldots, \mathbf{u}(n-1), x(n)\},$$

  where $\mathbf{u}(n)$ denotes the controls chosen by all players at period n (i.e., $\mathbf{u}(n) = (u_1(n), \ldots, u_p(n))$). A *policy* or a *strategy* is a way for each agent, to adapt his/her current control choice to the history of the system, that is a mapping $\gamma_i\colon (n, h_n) \to U_i$ which tells player i which control $u_i(n) \in U_i$ to select given that the time period is n and the state history is h_n.
- Once such a model is formulated the question arises as to what strategy or policy should each agent adopt so that his/her decision provides him/her with the most benefit. The decision to adopt a good strategy is based on a performance criterion defined over the life of the agent (in this case $[0, +\infty)$), that is, for each time horizon N the payoff to player i is determined by

  $$J_N^i(x, \mathbf{u}) = \sum_{n=1}^{N} \beta_i^n g_i(x(n), \mathbf{u}(n)),$$

  where x and $\mathbf{u}$ denote the state and control evolutions over time, $g_i\colon \mathbf{R}^{m_0} \times \cdots \times \mathbf{R}^{m_p} \to \mathbf{R}$ is a given reward function and $\beta_i \in [0, 1]$ is a discount factor for each player $i = 1, \ldots, p$.

Two categories of difficulties have to be dealt with when one studies infinite horizon dynamic games:

- the consideration of an unbounded time horizon gives rise to the possibility of having diverging values for the performance criterion (i.e., tending to $+\infty$ on all possible evolutions). This happens typically when there is no discounting ($\beta_i = 1$). A related issue is the stability vs. instability of the optimally controlled system.
- A second category of difficulties are associated with the consideration of all possible actions and reactions of the different agents over time, since an infinite time horizon will always give any agent enough time to correct his/her strategy choice, if necessary.

Triantaphyllou, E. and V.I. Torvik, (2001), "**Inference of Monotone Boolean Functions,**" *Encyclopedia of Optimization,* (P.M. Pardalos and C. Floudas, Eds.), Kluwer Academic Publishers, Boston, MA, U.S.A., Vol. 2, pp. 472-480.