Interactive Design Of Web Sites With A Genetic Algorithm

ARTICLE · FEBRUARY 2003

Source: CiteSeer

3 AUTHORS, INCLUDING:

Nicolas Monmarché
University of Tours
165 PUBLICATIONS **1,025** CITATIONS
SEE PROFILE

Gilles Venturini
University of Tours
200 PUBLICATIONS **1,535** CITATIONS
SEE PROFILE

INTERACTIVE DESIGN OF WEB SITES WITH A GENETIC ALGORITHM

A. Oliver, N. Monmarché, G. Venturini
Laboratoire d'Informatique, Université de Tours,
64, Avenue Jean Portalis, 37200 Tours, France.

ABSTRACT

We deal in this paper with the problem of automatically generating the style and the layout of web pages and web sites in a real world application where many web sites are considered. One of the main difficulty is to take into account the user preferences which are crucial in web sites design. We propose the use of an interactive genetic algorithm which generates solutions (styles, layouts) and which lets the user select the solutions that he favors based on their graphical representation. Two encodings have been defined for this problem, one linear and fixed length encoding for representing the style, and one variable length encoding based on HTML tables syntax for the layout. We present the results obtained by our method and the analysis of users behavior.

KEYWORDS

web site design, interactive genetic algorithms, style, layout, HTML

1. INTRODUCTION

The context of this study is a real world problem that arises in large-scale systems that are intended to host many web sites. In our case, a French company has developed such a software called DSP (Dynamic Site Publisher) which can be viewed as an advanced tool for numerous web sites design, maintenance and observation. It can be used from remote computers through web browsers in an intuitive way by users who may not need to be computer scientists.

This work focuses on the following problem: when creating or modifying a site, the user/customer may for instance specify the style of his site, and more precisely texts colors, fonts, etc. The user may set those elements by hand with a specific editor, or can use some predefined models of web site styles. The layout of a page is determined in a similar way (with predefined models for beginners and/or a more complex layout editor for advanced users). Editing those style or layout properties by hand can be time consuming and difficult for a standard user who may not be aware about HTML syntax and possibilities. Proposing predefined models is fine but it raises other problems, especially if thousands of web sites are considered: how many such predefined models should be proposed? And which predefined models should be proposed (since thousands of possibilities may exist)? So users of this kind of software may need an automatic tool to make them valuable creative suggestions in order to personalize the looking of their sites. This raises an additional problem which is how to automatically take into account the user's preferences in an intuitive and efficient way, i.e. without mathematical formulation and with an adaptation to each specific user.

For solving this problem, we propose a new method that interactively optimizes the look of web sites. We focus our study on the optimization of the site's style, i.e. text and background colors, fonts, etc, and on the optimization of the layout of web pages, i.e. the position of the different elements on a page. Our method uses an interactive genetic algorithm (Holland 1975) (Dawkins 1986): this algorithm generates solutions (i.e. styles or layouts) and lets the user select the individuals that he favors based on their graphical representations. Then the genetic algorithm takes into account the selected individuals in order to generate

the next population of possible solutions. In this way, the user drives in an intuitive and interactive manner the genetic search towards satisfying styles or layouts.

The remaining of this paper is organized as follows: section 2 describes the generic interactive genetic algorithm that we use for both style and layout optimization. In sections 3 and 4 we present the two respective encoding and genetic operators associated with the optimization of style or layout. In section 5 are described the results obtained on our demonstration site. Section 6 concludes on future work.

2. A GENERIC INTERACTIVE GENETIC ALGORITHM

2.1 What are interactive genetic algorithms?

Genetic algorithms (GAs) (Holland 1975) and more generally evolutionary algorithms (Spears et al. 1993) are stochastic search procedures which have been applied with success to many types of problems. Applications centered around the web are not yet as numerous as in other fields. They deal for instance with web mining and information filtering by evolving a population of agents or genetic individuals (Sheth 1994) (Menczer et al. 1995) (Morgan and Kilgour 1996) (Moukas 1997) (Fan et al 1999). In these studies an interactive relevance feedback process may occur between the agents and the user in order to better refine the user request. GAs have also been applied to web objects caching (Vakali and Manolopoulos 1999).

Interactive genetic algorithms (IGAs) are an extended version of GAs where the evaluation is performed by the user. This means that the user may either give a mark to the individuals, and these marks are used as an evaluation function, but he may more simply select those who will give birth to offspring. In both cases the user can drive the genetic search at his will and for instance according to his aesthetic preferences. The first IGA has been proposed by Dawkins in one of his book (Dawkins 1986) in order to highlight the power of natural selection and the accumulation of small changes in an evolutionary process. A survey of IGAs is beyond the scope of this paper and we recommend the interested reader a review of interactive evolutionary computation in (Takagi 2001) with more than 250 references, and a web site (Lewis 2000) where very interesting links are given.

In our previous work, we have applied IGAs to knowledge discovery in databases (Venturini et al. 1997). In (Monmarché et al. 1999) we have proposed a first prototype for style optimization only (with style sheets) which was however limited to style optimization and was not integrated in a real application.

2.2 Main algorithm

The optimization of the style and the layout are done separately, with different genetic representations and operators, but both make use of the same IGA described in this section. Separating these two optimization procedures was motivated by the fact that the user may want to optimize only one of the two aspects of his web site, and also to simplify the interactive evaluation of individuals. The generic IGA presented here is based on our previous experiences (Venturini et al. 1997) (Monmarché et al. 1999) which have highlighted important features that facilitate the interactive selection of individuals. This algorithm consists in the following steps:

1. Generate an initial population of N individuals with a random creation operator or with an existing style/layout
2. Display the individuals: the N styles (resp. layout) are applied to the user web page and the resulting N versions of the page are displayed (see for instance figure 1)
3. Possibly Stop: the user may instantaneously apply one satisfactory style/layout to his web page/site
4. Select individual(s): the user checks the individuals that he favors,
5. Generate a new population: keep the Nselect checked individuals for the next generation, discard the others and replace them with newly generated individuals according to:
    a. Nselect = 0: the whole population is regenerated (random creation as in step 1),
    b. Nselect = 1: N-1 new individuals are generated with the mutation (rate of Pmut),
    c. Nselect = N: the next generation is identical to the current one,

d. Nselect ∈ [2, N-1] : each of the (N-Nselect) new individuals are generated either by 1) (with probability Pcross) the crossover operator which is applied to two randomly selected parents among the checked individuals in order to create one offspring, and followed by the mutation operator applied to this offspring, or 2) (with probability (1-Pcross)) the mutation operator only which is applied to one randomly selected individual (among the checked ones),

6. Go to 2 in order to display the new generation.

Figure 1. Screen shot of an initial population obtained when optimizing the layout of a given web site

The population size N has been limited to 12 because it is important from the user point of view to display all the individuals on the same screen. The probability of mutation Pmut controls the diversity of the individuals and decreases over time from Pmax to Pmin.. In this way, the population will explore a lot of possibilities at the beginning of the search and then it will narrow around the user preferred individuals. In the following, we have experimentally set Pmin = 0.1, Pmax = 0.9, and Pcross = 0.6. Those values give the user the feeling that his choices are taken into account and nevertheless let the IGA explore a lot of possibilities.

3. STYLE OPTIMIZATION

Table 1. The 14 genes which encode the style of a page or site

Gene	Description	Values
Color1	color of the menu's text	(R, G, B) ∈ $[0,255]^3$
Color2	background color of menu	(R, G, B) ∈ $[0,255]^3$
Color3	first possible color of text	(R, G, B) ∈ $[0,255]^3$
Color4	background color for Color3	(R, G, B) ∈ $[0,255]^3$
Color5	second possible color of text	(R, G, B) ∈ $[0,255]^3$
Color6	background color for Color5	(R, G, B) ∈ $[0,255]^3$
Font	font for page's texts	9 fonts
Size	font size	[1,4]
Bold	style for page's texts	yes/no
Italic	"	yes/no
Underlined	"	yes/no
Alignment	"	centered/left/right
Bullet	paragraphs start with a bullet	yes/no
Borders	borders around images	yes/no

We have represented in table 1 the 14 genes that are used to encode the style of a page or site. Those values correspond to the parameters used by DSP in its database. Values for some genes like "Font" are parameterized, and this allows us for instance to add more fonts if needed.

The creation operator generates an individual by randomly choosing a value for each gene within its set of values. We have observed that random generation may produce individuals where the text is not readable because its color is too "close" to the corresponding background color. We have thus introduced a test on luminance (computed with the standard formula 0.2426 R + 0.7152 G + 0.0722 B). If any of the 3 couples (text color, background color) has a difference in luminance of less than 80, then the creation operator is run again.

The mutation consists in modifying each gene g with a probability Pmut. The crossover operator is a uniform crossover (Syswerda 1989) which generates one offspring D by randomly choosing each gene values of D either from one parent or the other with a probability of 0.5. Also, since colors can be easily mixed together, we have added the possibility to linearly combine the colors of the two parents. The luminance test is also applied to the individuals generated with mutation and crossover.

4. LAYOUT OPTIMIZATION

4.1 Genetic encoding

Table 2. The genes used to represent the layout of n objects in a HTML document

Gene	Description	Values
Cell 1		
X, Y	coord. of upper left corner	[1,n] ×[1,3]
Rowspan, Colspan	cell dimensions	[1,n] ×[1,3]
AlignH	horizontal alignment for objects in cell	[left, center, right]
AlignV	idem for vertical alignment	[top, center, down]
Color	cell background color	transparent, Color1, Color3, Color6 (see style)
Object1 to Object3	0 to 3 objects in the cell	$[0,n]^3$
Cell 2 (same genes as Cell 1)		
...		

We consider that the user wants to define a page layout for n objects denoted by O_1, ..., O_n. Those objects will be images or texts. The genetic encoding for the layout must represent the 2D organization of the objects. Several authors have already defined some representations for 2D structures in the context of bin-packing problems solved by GAs, like for instance (Schnecke 1996) (Dexter et al. 1997). However, the problem here is different because we do not try to optimize the size of the page and because we need to use a representation which is close to the HTML. The layout of a page is represented by an HTML table with a variable number of cells, where each cell may have different dimensions and may possibly contain up to 3 objects. Since cells can be merged together, a very important variety of layouts can be generated with such a representation. More precisely, an individual is mainly represented as a list of cells. The table containing those cells will have a dimension of at most n ×m, which means for instance that the n objects can be vertically aligned if necessary. In the following, the number of columns has been experimentally limited to m = 3 because it is more convenient to have long rather that wide HTML pages.

The resulting genetic representation is shown in table 2. Two types of constraints apply to this representation and are handled by the genetic operators described in the next section:

- table constraints: the cells must represent a correct HTML table (rectangular, no overlap between cells, overall dimensions included in [1..n]×[1..3], no empty rows or columns because HTML browsers may not correctly handle them),

- objects/cells constraints: each object must be included in a cell, and if a cell contains 1, 2 or 3 objects then these objects must be either one text, one image, an image and a text, or two images and a text. In this way, the genetic representation may encode the fact that a text is wrapped around one or two images, which is an important feature for web document layout.

4.2 Genetic operators

4.2.1 Creation

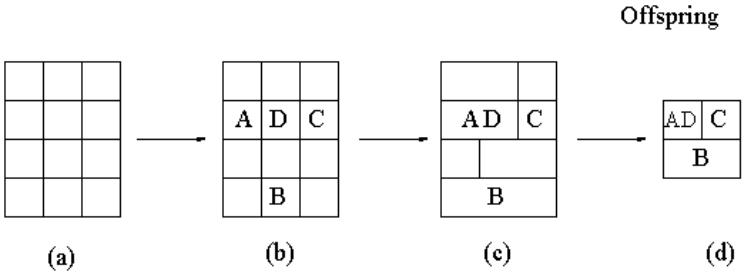

Figure 2. The creation operator that randomly generates layouts (with four objects A, B, C, D)

The creation operator generates one offspring D and works as follows:

1. D is initialized to a table of exactly n ×3 cells, (see figure 2(a)),
2. each object O_i is randomly assigned to a cell (see figure 2(b)),
3. AlignH, AlignV and Color genes of each cell are randomly generated and cells are enlarged by repeating merge operations (see figure 2(c)):
    a. for each cell C of coordinates X and Y, we compute how far it can be merged horizontally or vertically with the two following values: $Colspan_{max} = m - X + 1$ and $Rowspan_{max} = n - Y + 1$. Then for this cell, two desired values for Colspan and Rowspan are randomly generated within $[1,Colspan_{max}]$ and $[1,Rowspan_{max}]$ respectively,
    b. all cells are considered in a random order and we enlarge each of them according to the desired Colspan and Rowspan values. We start by any direction (vertical or horizontal) and enlarge the cell up to the desired value provided it does not violate objects/cells constraints (because when two cells are merged together, the resulting larger cell inherits of the objects contain in each smaller cell), and provided it does not violate table constraints (no overlap between cells).
4. empty rows or columns are eliminated (see figure 2(d)).

4.2.2 Mutation

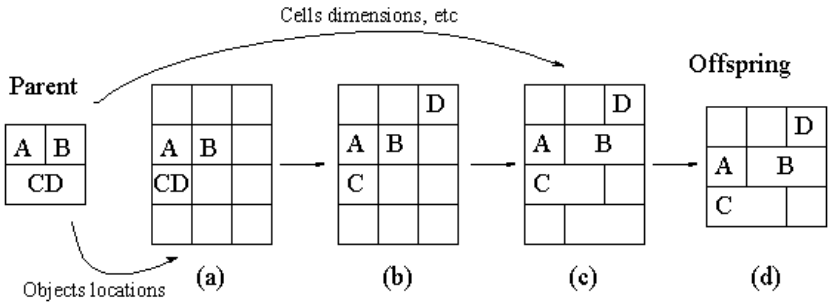

Figure 3. The mutation operator for layout optimization (with four objects A, B, C, D).

The mutation operator consists in generating an offspring D from one parent P with the same algorithm as in the creation operator, except that genes of P are used instead of completely random values. Those genes are possibly mutated. D is initialized to a n ×3 table (see figure 3(a)). Object locations are mutated (see figure 3(b)). Other genes are mutated, like for instance the desired values for Rowspan and Colspan. Cells are

possibly enlarged according to the desired Colspan or Rowspan values as in the creation operator. Empty rows or columns are eliminated (see figure 3 (d)).

4.2.3 Crossover

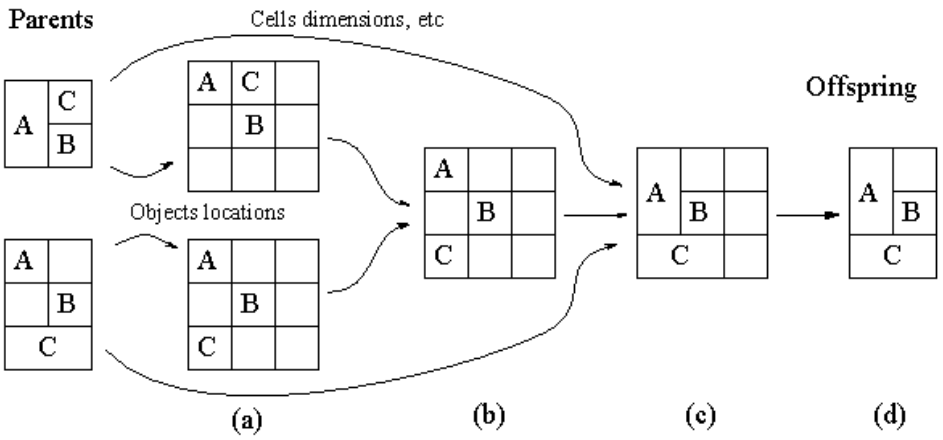

Figure 4. The crossover operator for layout optimization (with three objects A, B, C)

The crossover operator combines the layouts represented by two parents P_1 and P_2 in order to create one offspring D:

1. D is initialized to an empty n×3 table, and P_1 and P_2 are centered on a n×3 table (see figure 4 (a)),
2. objects are placed in D at the location given by P_1 or P_2 (see figure 4 (b)),
3. other genes of D are inherited either from P_1 or P_2, like for instance the desired values for Rowspan and Colspan. Then cells are enlarged as in the creation operator (see figure 4 (c)).
4. empty rows or columns are eliminated (see figure 4 (d)).

5. RESULTS

The system described in this paper is operational and has been integrated into DSP. When the user wants to define or modify the style/layout characteristics of his site (or of a given page), he may use the IGA among the other possible options (manual edition, predefined models). Once he has found satisfying results, he may apply instantaneously the style/layout to his site or page. This represents one of the first real world application of IGAs (see also (Herdy 1997)).

Figure 5 shows typical examples obtained with 2 texts and 3 images. Many different kinds of styles/layouts can be generated according to the user preferences. Results are thus very often subjective. A demo version of this system (limited to style optimization) can be tested on the "demo" page of our web site: http://www.antsearch.univ-tours.fr/webrtic/.

We do not have statistics about the DSP users/customers because analyzing log files or tables for site administration sessions is difficult. However, we can analyze sessions on a demonstration site that we have maintained for a few months. From a hundred of such sessions, we have selected 51 of them based on their representativeness (at least one generation of individuals generated). Among those sessions, 33 of them concern style and layout optimization, 11 sessions concern style optimization only, and 7 sessions concern layout optimization only. The mean length of a session is 9 minutes (with standard deviations of 11 minutes). The mean number of generations for style and layout optimization are respectively 7.2 and 6.1. The average time needed from one generation to the next one is 48.9 seconds. About half of this time is devoted to the network and to the display. These data confirm that this tool is intuitive and easy to use because it allows the user to find satisfying results in few minutes. People who have tested this demo were the company's employees (27%), other people from France (64%), and people from foreign countries (9%).

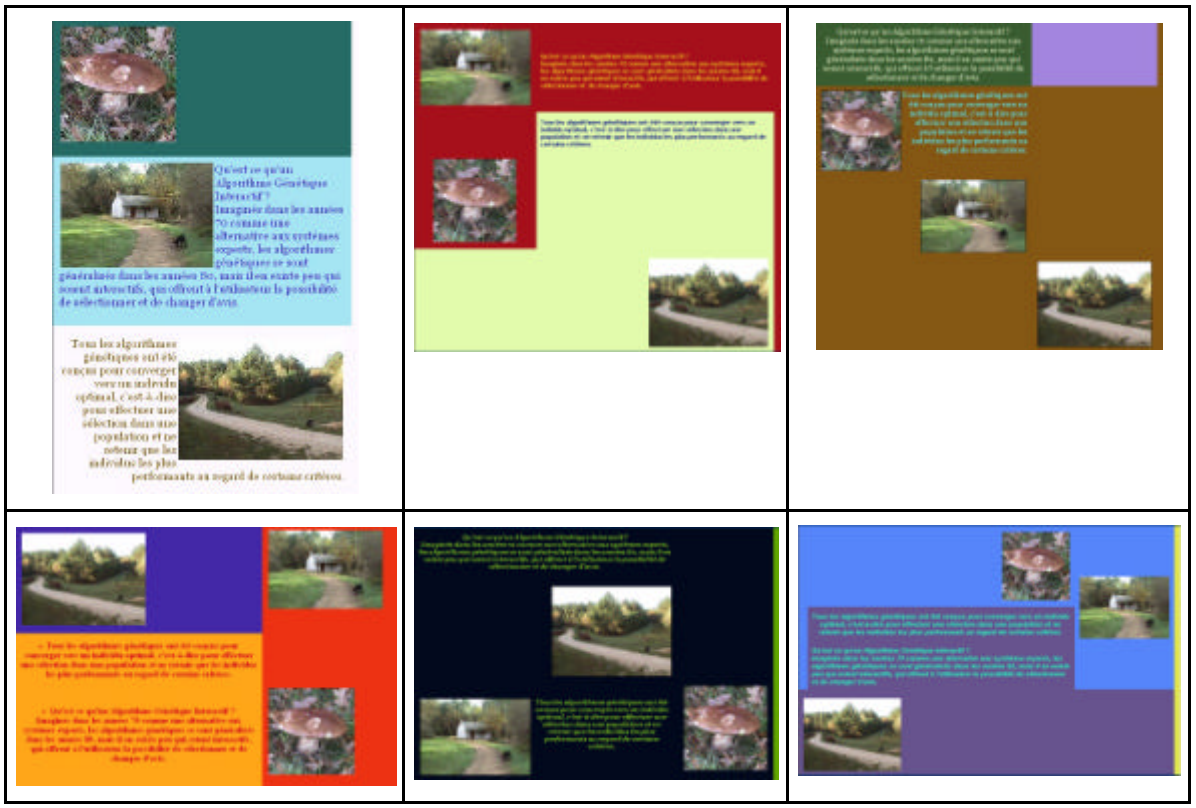

Figure 5. Typical results obtained with 2 texts and 3 images. This illustrates the variety of styles/layouts which can be generated with our tool.

6. CONCLUSION AND FUTURE WORK

We have presented in this paper how to use interactive genetic algorithms in web site design. The main points in our work are the followings: 1) the genetic encoding which has been defined can represent millions of possible styles and layouts 2) the interactive process in which the user selects the individuals that he favors is simple and very intuitive, 3) the genetic search is performed by adapted genetic operators which result in good performances according to user behavior on our demonstration site. Furthermore, it has been integrated in a real world application.

One important feature that could be taken into account in a short time is the possible dependencies that may exist between elements (like for instance, this text must close to this image and far from this other text). This can be done quite easily thanks to the GA because it would not change at all the genetic operators but simply the evaluation of individuals. An evaluation function could be defined by looking at each object in the page and by penalizing/rewarding the layout each time the neighborhood of this object contains an undesirable/desirable object. This evaluation function would eliminate individuals that would not reach a given level of quality. We have not implemented yet such mechanism because we think that it would be better suited for advanced users rather than beginners. But it is a very interesting issue which raises also the problem of using within IGAs an existing evaluation function in conjunction with the user interactive evaluation.

REFERENCES

Dawkins R. (1986), *The Blind Watchmaker*, Longman, Harlow, 1986.

Dexter T., Goodman E.D. , Punch W.F. (1997), The genetic algorithm and local optimizer hybrid approach for the advanced layout problem. GARAGe97-02-03 Technical Report, Michigan State University, Feb 97.

Fan W., Gordon M.D., Pathak P. (1999), Automatic generation of a matching function by genetic programming for effective information retrieval, Proceedings of the 1999 Americas Conference on Information Systems, pp 49-51.

Herdy M. (1997), Evolutionary optimization based on subjective selection - Evolving blends of coffee, Proceedings of the 5th European Congress on Intelligent Techniques and soft Computing, EUFIT'97, 1997, pp 640-644.

Holland J.H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

Lewis M. (2000), Visual aesthetic evolutionary design links, http://www.cgrg.ohio-state.edu/~mlewis/aed.html

Menczer F., Belew R.K., Willuhn W. (1995), Artificial life applied to adaptive information agents, Spring Symposium on Information Gathering from distributed, Heterogeneous Databases, AAAI Press, 1995.

Monmarché N., Nocent G., Slimane M. and Venturini G. (1999), Imagine: a tool for generating HTML style sheets with an interactive genetic algorithm based on genes frequencies. 1999 IEEE International Conference on Systems, Man, and Cybernetics (SMC'99), Interactive Evolutionary Computation session, October 12-15, 1999, Tokyo, Japan.

Morgan J.J., Kilgour A.C. (1996), Personalising information retrieval using evolutionary modeling, Proceedings of PolyModel 16: Applications of Artificial Intelligence, ed by A.O. Moscardini and P. Smith, 142-149, 1996.

Moukas A. (1997), Amalthea: information discovery and filtering using a multiagent evolving ecosystem, *Applied Artificial Intelligence*, 11(5):437-457, 1997

Schnecke V. (1996), Hybrid Genetic Algorithms for Solving Constrained Packing and Placement Problems, PhD. Thesis, University of Osnabruck, Department of Mathematics/Computer Science, December 1996

Sheth B.D. (1994), A learning approach to personalized information filtering, Master's thesis, Department of Electrical Engineering and Computer Science, MIT, 1994.

Syswerda G. (1989), Uniform crossover in genetic algorithms, Proceedings of the third International Conference on Genetic Algorithms, 1989, J.D. Schaffer (Ed), Morgan Kaufmann, pp 2-10.

Takagi H. (2001), Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation, to appear in *Proceedings of the IEEE*, 2001, 22 pages.

Vakali A., Manolopoulos Y. (1999), Caching objects from heterogeneous information sources, Technical report TR99-03, Data Engineering Lab, Department of Informatics, Aristotle University, Greece.

Venturini G., Slimane M., Morin F. and Asselin de Beauville J.-P. (1997), On using interactive genetic algorithms for knowledge discovery in databases, Proceedings of the seventh International Conference on Genetic Algorithms, 1997, T. Baeck (Ed.), Morgan Kaufmann, pp 696-703.