WIRTSCHAFTSWISSENSCHAFTLICHES ZENTRUM (WWZ)
DER UNIVERSITÄT BASEL

March 2004

Development of an Intelligent Tutoring System for AHP (Analytic Hierarchy Process)

Alessio Ishizaka

WWZ Forum, Petersgraben 51, CH - 4003 Basel
Fax +41 61 267 33 33 | e-mail: forum-wwz@unibas.ch | www.wwz.unibas.ch/forum

The Author:

Alessio Ishizaka was researcher at the Department for Information Systems of the University of Basel, Department of Business and Economics (WWZ).

For further infomation please contact:

Prof. Dr. oec. Markus Lusti, Economic Theory
markus.lusti@unibas.ch

A publication of the University of Basel, Department of Business and Economics (WWZ)
Address for information:
WWZ Forum, Petersgraben 51, CH-4003 Basel Fax +41 61 267 33 33
The WWZ Research Papers document the results of research projects at the WWZ.

TABLE OF CONTENTS

Table of figures

Résumé

Ce travail considère, décrit et discute l'implémentation d'AHP-Tutor, un Système Tutoriel Intelligent (STI) enseignant divers concepts et méthodes de dérivation des priorités d'une matrice des comparaisons d'AHP. Le but de notre système est de proposer des exercices. La théorie correspondante doit être acquise de façon traditionnelle en classe, par des livres ou par un enseignement assisté par ordinateur conventionnel.

A la suite d'une recherche bibliographique, nous avons retenu cinq méthodes qui seront implémentées dans notre système:

- la méthode de la valeur propre dominante droite
- la méthode de la valeur propre dominante gauche
- la méthode de la moyenne des valeurs normalisées
- la moyenne géométrique des lignes
- la moyenne géométrique des colonnes

Une étude comparative de ces méthodes sur 500 matrices inconsistantes ne relève aucune différence majeure dans le classement des solutions.

Basé sur ces recherches, un curriculum de trois types d'exercices a été élaboré. Chaque exercice a un style d'enseignement différent: guidage strict, résolveur et découverte libre. Le choix du style a été fixé par les objectifs à enseigner.

AHP-Tutor a été développé avec Visual Prolog 5.2 et fonctionne sous MS Windows. L'accent a été mis sur le module expert qui est la colonne vertébrale d'un système tutoriel intelligent. Sa composante explicative présente à l'étudiant les étapes conduisant à la réponse. Elle reprend le raisonnement du système pour l'adapter à l'élève. Cette composante est indépendante du domaine.

Abstract

This work describes and discusses the implementation of AHP-Tutor, an Intelligent Tutoring System (ITS) that provides training in various methods of derivation of priorities from an AHP comparison matrix. The purpose of our system is to offer exercise problems. The corresponding theoretical knowledge has to be acquired by other means, for example in the classroom, from books or through conventional computer-assisted learning.

Based on bibliographical research, five methods were selected for implementation:

- the method of the maximal right eigenvalue
- the method of the maximal left eigenvalue
- the method of the mean of the normalized values
- the geometric mean of the rows
- the geometric mean of the columns.

A comparative study applying these methods to 500 inconsistent matrices did not reveal any major difference in the classification of alternatives.

Based on these findings, a curriculum consisting of three types of exercises was developed. Each exercise uses one of the following three teaching styles: strict guidance, solver, or free discovery. The choice of style was fixed according to the teaching objectives.

AHP-Tutor was developed with Visual Prolog 5.2 and runs under MS Windows. The main emphasis is on the expert module, which is the backbone of an intelligent tutoring system. Its explanation component shows the student the steps leading to the solution. It reuses the reasoning of the system and adapts it to the student. This component is domain-independent.

Chapter I

Introduction

In a class of students some solve the exercises quickly and with few errors, other display more difficulties. Solving the exercise on the blackboard shows the correct solution, but generally does not allow the students to locate their mistakes. Better corrections are made when the teacher follows a student individually.

Intelligent Computer Assisted Learning, offering the permanent availability of an artificial teacher, makes it possible to assist students individually. It corrects erroneous calculations immediately. Thus, it offers to each student an adaptable learning environment.

This work is interested in the implementation of an Intelligent Tutoring System on the theoretical basis of the decision method *Analytic Hierarchy Process* (AHP). In our University, AHP is taught in the fifth semester. Each of four theoretical lessons is accompanied by a training lesson which makes it possible to acquire practical knowledge which cannot be learned passively by books or by listening to a teacher. The student is confronted with problems, makes mistakes, and experiences dead ends to reveal gaps in his knowledge.

The first four lessons are devoted to the application of AHP. Among these, two use the commercial software *Expert Choice*. The knowledge taught helps to use the supporting software, but is unaware of how the results are calculated. It is not enough to rationally evaluate the method and can even lead to the inappropriate use of the software. Therefore, the four other lessons teach the theoretical basis of AHP. The first two cover the paragraph 2.5 of the text book [LUSTI02, p. 32-43]. The two subsequent ones are practical. They illustrate the theory by means of the intelligent tutoring system *AHP-Tutor* developed in this project.

After this introduction, chapter 2 reminds the basis of AHP. In chapter 3 five priorities derivation methods are compared in a Monte Carlo simulation. In chapter 4, the curriculum of AHP-Tutor is described. Chapter 5 introduces its architecture, and chapter 6 exposes the explanation component.

Chapter 2

AHP

2.1 Introduction

A complex world needs complex decisions. To assist in this difficult task, decision support methods have been developed. They try to supersede choices based on experience or intuition and allow decisions based on scientifically based arguments. A decision support method tries to model the preference system in the mind of the decision-maker.

According with the nature of the problem, different families of methods have been proposed. The *Analytic Hierarchy Process* (AHP) is a multicriterial method which tries to satisfy several conflicting criteria. More precisely, it belongs to the Multi-attribute Decision Making (MADM) field because the decision space is discrete[1] [for example HWANG81, p. 1-4; ZIMME96, p. 303-304].

There are different methods of MADM, which can be classified by different criteria, for example: the kind of data used or the number of people implied in the decision [TRIAN00, p.3-4].

[CHEN92, p.20-23] gives a classification which is widely recognised by the scientific community. It is based on two criteria (see figure 2.1):

- Is information required by a decision-maker?
- Which kind of decision is required?

[MAYST94, p.16-17] introduces a supplementary ramification for the methods needing cardinal information: the methods with full aggregation and those with semi aggregation. The methods with full aggregation consider an optimum and the classification of alternatives is always possible. Methods with semi aggregation accept the possibility of incomparability and thus of intransitivity. Only certain

[1] Problems with continue decision space belong to the Multi-Objective Decision Making (MODM) filed

elements are brought out. If an optimum does not exist, one prefers to present a subset of alternatives instead of imposing one.

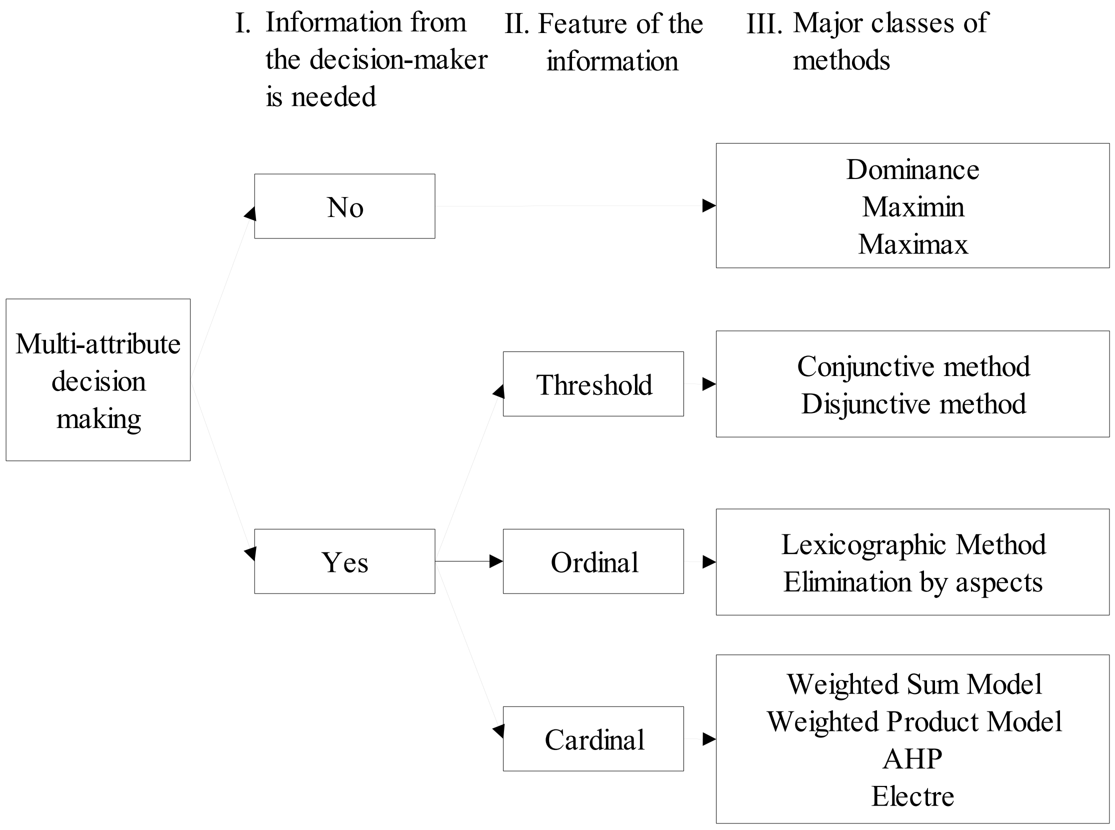


Figure 2.1: Taxonomy of MADM Methods (according to [CHEN92, p.23])

Each multi-attribute decision process with cardinal information (figure 2.2) is based on four steps:

a) Determine the goal, the criteria and the alternatives.
b) Allocate values to criteria representing their relative importance and deliver values to alternatives representing their effects on these criteria.
c) Process the values to determine a classification of the solutions.
d) Analyse the sensibility of the results.

[TRIAN00, p.5] mentions only the first three steps but every manual describing a decision method will recommend closing the process with a sensibility analysis. It systematically varies the values allocated to the different parameters of the method to assess the uncertainty of the derivation.

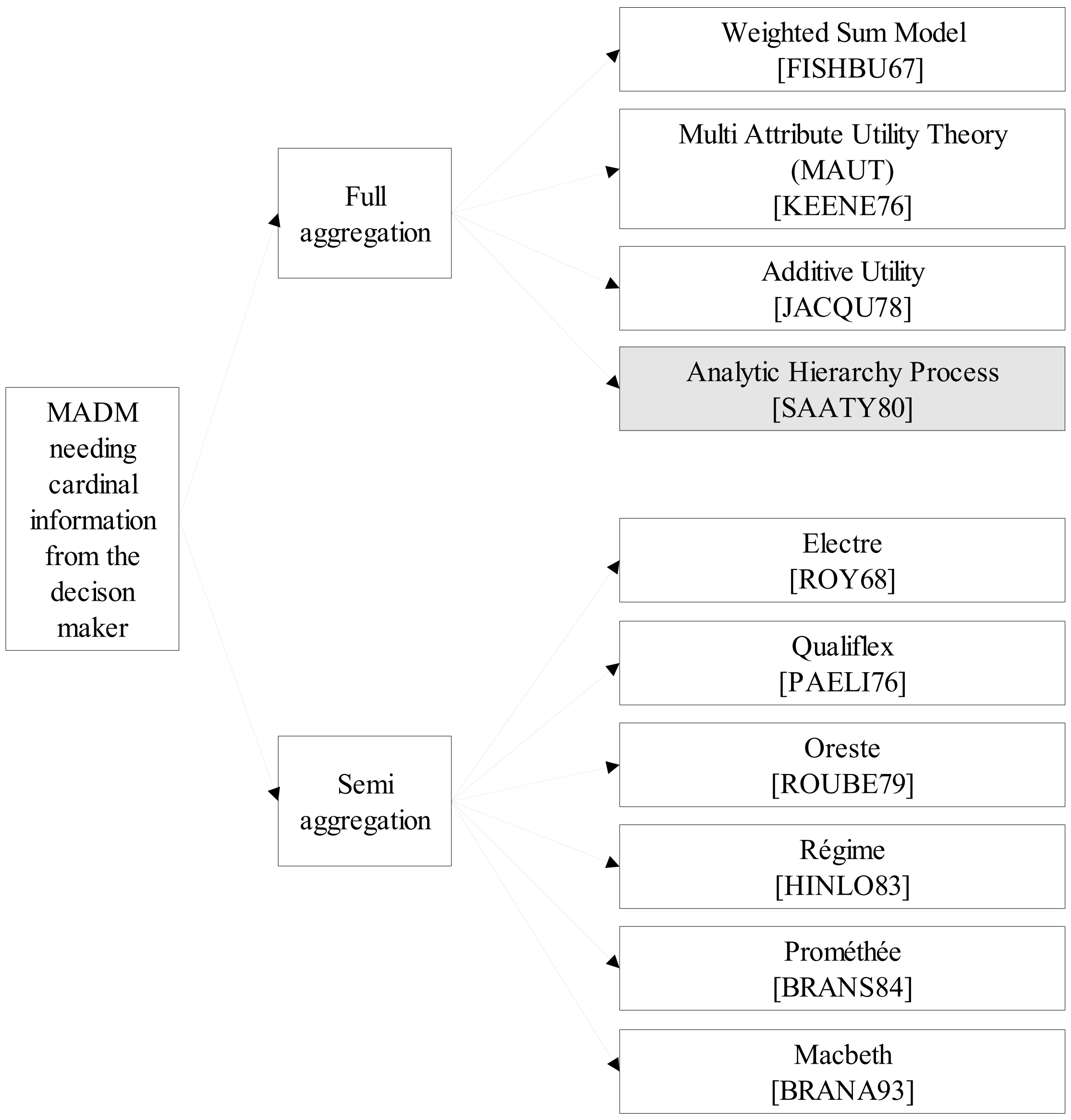

MADM needing cardinal information from the decison maker
Full aggregation
Semi aggregation
Weighted Sum Model [FISHBU67]
Multi Attribute Utility Theory (MAUT) [KEENE76]
Additive Utility [JACQU78]
Analytic Hierarchy Process [SAATY80]
Electre [ROY68]
Qualiflex [PAELI76]
Oreste [ROUBE79]
Régime [HINLO83]
Prométhée [BRANS84]
Macbeth [BRANA93]

Figure 2.2: Taxonomy of MADM methods needing cardinal information [MAY-ST94, p.17].

2.2 Comparison Matrix

2.2.1 Consistency

The Analytic Hierarchy Process (AHP) evaluates decision alternatives by pair wise comparisons, thus allowing more accurate judgements than the simple weighted product model [SAATY94, p.8-9]. The comparisons are recorded in a matrix (figure 2.3). This matrix is square and reciprocal.

A =

1	...	a_{ij}	...	a_{1n}
...	1	...	...	...
$1/a_{ij}$	...	1	...	...
...	...	...	1	...
$1/a_{1n}$	...	....	...	1

Figure 2.3: Comparison matrix

The elements a_{ij} of a comparison matrix **A** compare the alternatives i and j of a decision problem. The a_{ij} are said to be consistent if they respect the following transitivity (2.1) and reciprocity (2.2) rules (examples 2.1 and 2.2):

(2.1) $a_{ij} = a_{ik} \cdot a_{kj}$ where i, j and k are any alternatives of the matrix

Example 2.1:
Suppose you like an apple twice as much as an orange (a_{12} = 2) and an orange three times as much as a banana (a_{23} = 3). If you like an apple six times as much as a banana (a_{13} = 6), the transitivity rule is respected.

(2.2) $a_{ij} = \frac{1}{a_{ji}}$

Example 2.2:
If you like an apple twice as much as an orange (a_{12} = 2), then you like an orange half as much as an apple (a_{21} = 1/2).

A comparison matrix is reciprocal because its inferior part is reciprocal to the superior part and all the elements of the principal diagonal are 1. Therefore a transitivity test of one of the two parts of the matrix is sufficient:

(2.3) $a_{ij} = a_{ik} \cdot a_{kj}$ for $j > k > i$

Hence, for each element a_{ij} a number of j-(i+1) equations (2.3) have to be respected (example 2.3).

Example 2.3:
The comparison a_{25} has to conform to the equations $a_{25} = a_{23} \cdot a_{35}$ and $a_{25} = a_{24} \cdot a_{45}$.

These equations can be expressed only by terms of the first diagonal above the principal diagonal (i.e. $a_{12}, a_{23}, \ldots, a_{n-1n}$) (example 2.4):

(2.4) $\quad a_{ij} = a_{ii+1} \cdot a_{i+1i+2} \cdot \ldots \cdot a_{j-1j}$

Example 2.4:

$a_{25} = a_{23} \cdot a_{35}$

$\boxed{a_{35} = a_{34} \cdot a_{45}}$

$a_{25} = a_{23} \cdot a_{34} \cdot a_{45}$

An n x n matrix contains n^2 comparisons. n-1 of the elements can be chosen freely, the other elements have to consider the reciprocity (2.2) and transitivity rules (2.4).

2.2.2 How to Build a Consistent Matrix

A *step-by-step* method checks the consistency of each comparison entered [ISHIZ04a]. This method is more efficient than filling in the comparison matrix completely and then reconsidering the comparisons for better consistency. This method consists of four phases which correspond to the types of comparisons: comparisons on the principal diagonal, independent comparisons, transitive comparisons and reciprocal comparisons (figure 2.4).

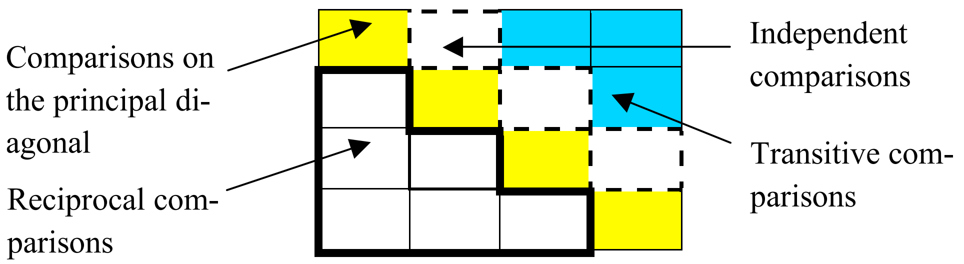


Figure 2.4: A matrix with the different types of comparisons

1) The principal diagonal contains all comparisons of an alternative with itself (figure 2.5):

1				

	1			
		1		
			1	
				1

Figure 2.5: Principal diagonal of a comparison matrix.

2) Independent comparisons are not linked to other comparisons by the transitivity or reciprocity rules. This is the case if the first *n-1* comparisons are distributed over all columns or rows respectively (counter example in figure 2.6).

1	0.33	0.5	4	
	1		8	
		1		
			1	
				1

Figure 2.6: Instead of a single comparison the fourth column contains two entries. This way, the first *n-1* comparisons are dependent, because of $a_{14} = a_{13} \cdot a_{24}$ (transitivity rule). To keep the comparisons independent each entry has to be in a new column.

Beginning with the first row (a_{12}, a_{13}, …,a_{1n}) is questionable. It can be argued that comparing alternatives linewise compromises the (psychological) independence of the comparisons, an advantage of the pairwise AHP method compared to the simultaneous procedure. We therefore choose the first diagonal as a starting point (figure 2.7). Furthermore, this choice allows the calculation of the other comparisons of the upper matrix by multiplication instead of division ($a_{ik} = \frac{a_{ij}}{a_{kj}}$).

1	0.5			
	1	2		
		1	1	
			1	0.25
				1

Figure 2.7: Choosing the first diagonal above the principal diagonal as a starting point.

3) Transitive comparisons can be deduced from the first diagonal entered in the second step (2.4) (figure 2.8). The comparison between alternative 1 and alternative 5, for example, is given by: $a_{15} = a_{12} \cdot a_{23} \cdot a_{34} \cdot a_{45} = 0{,}5 \cdot 2 \cdot 1 \cdot 0{,}25 = 0{,}25$.

1	0.5	1	1	0.25
	1	2	2	0.5
		1	1	0.25
			1	0.25
				1

Figure 2.8: The upper part of the first diagonal is deduced by the transitivity rule.

4) Each entry in the lower part of the comparison matrix is the reciprocal of the corresponding upper part entry (2.2). In figure 2.9, for example, a_{51} is $\frac{1}{a_{15}}$.

1	0.5	1	1	0.25
2	1	2	2	0.5
1	0.5	1	1	0.25
1	0.5	1	1	0.25
4	2	4	4	1

Figure 2.9: The lower part is the reciprocal of the upper part.

2.2.3 Inconsistency Tolerance

For the construction of a consistent matrix only the second step is necessary (section 2.2.2). The first step is trivial; the last two steps can be deduced by the transitivity (2.4) and the reciprocity rules (2.2). This was proposed in [WEDLE93] for the calculation of missing comparisons.

This procedure is not reliable for the transitive comparisons (step three) because it introduces values which are not necessarily present in the decision maker's mind. For instance, a weak preference value of 3 (on the fundamental scale) for the comparisons A vs. B and B vs. C would imply a predominant or absolute value for A vs. C of 9 (= 3 · 3).

To overcome this problem, our system lets the user choose the comparison scale and the error tolerated in the translation rule:

- The fundamental scale one to nine has shortcomings [DYER90]. For some problems a wider scale (e.g. one to fifteen) and for others a shorter scale is more appropriate. A description of various scales can be found in [TRIAN00, p.23-45].
- To allow a certain degree of inconsistency, a tolerated error *e* (percentage of the height value of the comparison scale) is introduced [ISHIZ04a]. Hence, the transitivity rule (2.4) is supplemented by an error term:

(2.5) $$a_{ij} = a_{ii+1} \cdot a_{i+1i+2} \cdot \ldots \cdot a_{j-1j+1} \pm \frac{e \cdot h}{100}$$

where *e* is the tolerated error

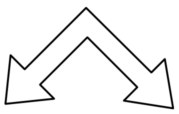
h is the height value of the comparison scale

If the extended transitivity rule (2.5) is violated, the user is required to modify either the value entered (left part of (2.5)) or the comparisons of the first diagonal (right part of (2.5)). Modifying the values of the first diagonal induces changes on the transitive comparisons. The user is offered both matrices and can adopt the most appropriate (example 2.5).

Example 2.5:
Suppose the user chooses 4 for the comparison a_{14} of the matrix of figure 2.10 and 0 for the tolerated error. But according to the transitivity rule (2.5) the user is supposed to enter $a_{14} = 0.5 \cdot 2 \cdot 1 = 1$. If the user maintains his entry, he can modify the comparisons a_{12}, a_{23}, and a_{34} of the first diagonal and see the effects on the comparisons above the first diagonal. Then he can either adopt the solution proposed by the program (figure 2.11) or proceed with his own matrix (figure 2.12).

1	0.5	1	**?**	
	1	2	2	
		1	1	0.25
			1	0.25
				1

Figure 2.10: Comparison matrix. The comparison a_{14} is requested.



1	0.5	1	1	

	1	2	2	
		1	1	0.25
			1	0.25
				1

Figure 2.11: Matrix with proposed comparison ($a_{14} = 1$).

1	**1**	2	4	
	1	2	4	
		1	**2**	0.25
			1	0.25
				1

Figure 2.12: Matrix after changing premises ($a_{12} = 1$ and $a_{32} = 2$).

2.3 Conclusion

This chapter showed how to classify the various methods of decision support. We learned that AHP is a method needing information from the decision maker on a cardinal scale and having a full aggregation.

The second part of the chapter described a method to build a consistent or a near consistent matrix. This method is more efficient than filling in the comparison matrix completely and then reconsidering the comparisons for better consistency.

The next chapter describes the different methods for deriving the priorities and comparing them in a Monte Carlo simulation.

2.4 Summary

♦ Decision support systems help decision makers in the choice of a solution for a complex problem.

♦ AHP needs information of the decision maker on a cardinal scale and have a full aggregation.

♦ The multi-attribute decision process is based on four steps:

a) Determine the goal, the criteria and the alternatives.

b) Allocate values to criteria representing their relative importance and deliver values to alternatives representing their effects on these criteria.

c) Process the values to determine a classification of the solutions.

d) Sensibility analysis.

- ♦ The Analytic Hierarchy Process (AHP) evaluates decision alternatives by pair-wise comparisons, thus allowing more accurate judgements than the simple weighted product model.

- ♦ A comparison matrix is consistent if all the comparisons obey the reciprocity rule $a_{ij} = \frac{1}{a_{ji}}$ and the transitivity rule $a_{ij} = a_{ik} \cdot a_{kj}$.

- ♦ The construction of a consistent matrix can be decomposed in four steps corresponding to the various types of comparisons: comparisons of the principal diagonal, independent comparisons, transitive comparisons and reciprocal comparisons.

- ♦ The redundancy of the information in the matrix allows logically inconsistent comparisons, which nevertheless may be psychologically valid. For this purpose, a tolerated error is introduced in the transitivity rule: $a_{ij} = a_{ii+1} \cdot a_{i+1i+2} \cdot \ldots \cdot a_{j-1j+1} \pm \frac{e \cdot h}{100}$ where e is the tolerated error and h is the height value of the comparison scale.

Chapter 3

Simulation of Priority Derivation

3.1 Introduction

Several methods have been developed to derive priorities from AHP matrices. They can be divided in two groups [GOLAN93]: the eigenvalue approach and the methods minimizing the distance between the user-defined matrix and the nearest consistent matrix.

Among the eigenvalue methods, we distinguish the principal right eigenvalue [SAATY77], [SAATY80], the principal left eigenvalue [JOHNS79] and the modified eigenvalue method. Because of the reciprocity of the matrix, the last utilizes only the upper triangle to calculate the priorities. Unfortunately, the ranking depends on the order of the alternatives in the matrix [COGGE85], [TAKED87].

A minimum distance can be reached by different metrics. This has lead to the development of different derivation methods (figure 3.1), in particular the logarithmic least squares also called geometric mean [CRAWF85], the least squares [JENSE84], the weighted least squares [CHU79], [BLANK87], and the logarithmic least absolute values [COOK88].

Method	Minimum of:
Logarithmic least squares method	$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(\ln(a_{ij}) - \ln(\frac{p_i}{p_j}) \right)^2$
Least squares method	$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} - \frac{p_i}{p_j} \right)^2$
Weighted least squares	$\sum_{i=1}^{n} \sum_{j=1}^{n} \left(a_{ij} \cdot p_j - p_i \right)^2$
Logarithmic least absolute values	$\sum_{i=1}^{n} \sum_{j=1}^{n} \left\| \frac{1}{2} - a_{ij} - \ln p_j + \ln(p_i) \right\|$

Figure 3.1: Derivation methods based on the minimization of the distance, p_i is the researched priority of alternative i, a_{ij} is the comparison between alternative i and j and n is the dimension of the comparison matrix

With the exception of the logarithmic least squares equation, the methods are difficult to apply. In particular, the least squares method has several minima and makes the choice ambiguous. Saaty [SAATY84a] gives an example where the least squares method even produces an illogical result.

A heated discussion has arisen over the "best" method. One side supports the eigenvalue method [SAATY84a], [SAATY84b], [HARKE97], [SAATY01], [SAATY03], the other side argues for the geometric mean [BARZI87], [BARZI90], [BARZI97], [BARZI01]. This dispute seems to be futile because experimental studies [BUDES86], [GOLAN93] show that each method is best in at least one criterion (usually among the criteria it explicitly seeks to optimize), but neither is optimal by all or even most criteria.

This chapter describes a Monte Carlo simulation studying the differences between five derivation methods: the right eigenvalue method, the left eigenvalue method, the geometric mean of the rows, the geometric mean of the columns and the mean of the normalised values. Other experimental studies [BUDES86], [GOLAN93] have investigated the "best" derivation method. Our study tries to show under which conditions the rankings of the above methods are similar and which factors influence a ranking contradiction. The first part of the chapter reviews the five methods theoretically. Then we describe a Monte Carlo simulation comparing the methods, and finally we evaluate the results.

3.2 Derivation of Priorities

In a perfectly consistent matrix, all the comparisons a_{ij} obey the equality $a_{ij} = \frac{p_i}{p_j}$, where p_i is the priority of the alternative i. Each method satisfies this propriety and calculates identical priorities for consistent matrices. To deal with inconsistencies two theories are proposed: the perturbation theory and the distance minimization.

3.2.1 Mean of the Normalized Values

This is the oldest method and is based on three steps (example 3.1):

1. Sum of the elements of the column j:

2. Normalization of the column *j*
3. Mean of row *i*

<u>Example 3.1</u>:

Consider the following comparison matrix:

1	6	3
1/6	1	1/2
1/3	2	1

The method "mean of the normalized values" derives the priorities as follow:

1. Add the elements of the columns: (1.5, 9, 4.5)
2. Normalize the columns

0.67	0.67	0.67
0.11	0.11	0.11
0.22	0.22	0.22

3. Calculate the mean of the rows: (0.67, 0.11, 0.22)

In the case of inconsistent matrices, this method cannot be mathematically justified.

3.2.2 The Eigenvalue Approach

[SAATY77], [SAATY80] proposes the principal eigenvector $\vec{p}$ as the desired priorities vector. It is derived from the following equation:

(3.1) $\mathbf{A} \cdot \vec{p} = \lambda \cdot \vec{p}$ where **A** is the comparison matrix

$\vec{p}$ is the priorities vector

λ is the maximal eigenvalue

Saaty justifies the eigenvalue approach for slightly inconsistent matrices with the perturbation theory, which says that slight variations in a consistent matrix imply slight variations of the eigenvector and the eigenvalue (example 3.2).

<u>Example 3.2</u>:
The figure 3.2 represents the characteristic equation of the consistent matrix **A**. The maximal eigenvalue is 3 (dimension of the matrix) and the associated eigenvector is $\vec{p}$ = (0.67, 0.11, 0.22). The figure 3.3 draws the characteristic equation of

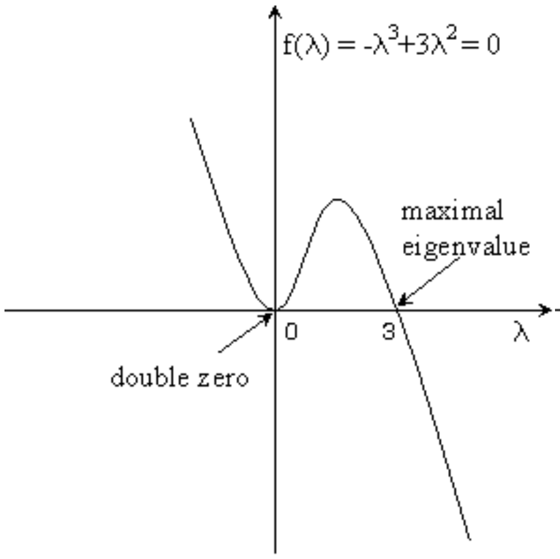
the near consistent matrix **B**, slightly modified from the matrix **A**. We can see that the characteristic equation, the eigenvalues and the priorities are also slightly modified.

consistent matrix

$$\mathbf{A} = \begin{array}{|c|c|c|}\hline 1 & 6 & 3 \\ \hline 1/6 & 1 & 1/2 \\ \hline 1/3 & 2 & 1 \\ \hline \end{array}$$

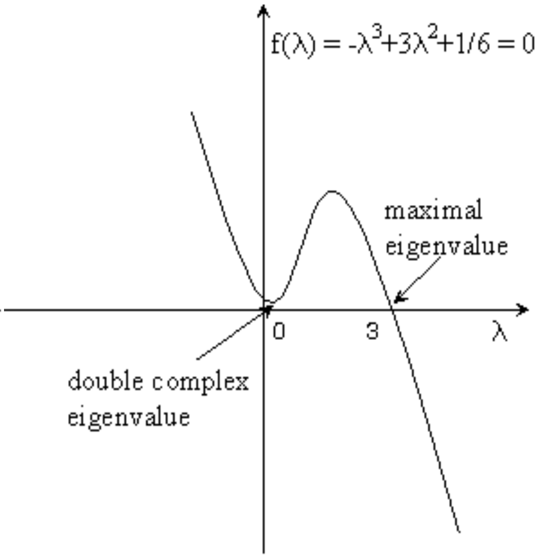
near consistent matrix

$$\mathbf{B} = \begin{array}{|c|c|c|}\hline 1 & 6 & \mathbf{2} \\ \hline 1/6 & 1 & 1/2 \\ \hline \mathbf{1/2} & 2 & 1 \\ \hline \end{array}$$



$\vec{p} = (0.67, 0.11, 0.22)$

Figure 3.2: Characteristic equation of the consistent matrix **A**



$\vec{p} = (0.61, 0.12, 0.27)$

Figure 3.3: Characteristic equation of the near consistent matrix **B**

The eigenvalue method is less transparent than the minimization of the distance. Many authors have underlined the lacking clarity of the eigenvalue process [JOHNS79], [CHU79]. To clarify the eigenvalue method we interpret the power method, a numerical method to calculate the maximal eigenvector (e.g [LUSTI02]).

The power method defines an iterative process:

1. The pairwise matrix is squared
2. The row sums are then calculated and normalised, it is the first approximation of the eigenvector
3. Using the last resulting matrix, repeat step 1 and 2

4. Step 3 is repeated until the difference between these sums in two consecutive priorities calculations is smaller than the stop criterion

<u>Example 3.3</u>:

Considering the inconsistent matrix B of figure 3.3, the priorities are derived as follows:

1. Square the matrix

3	16	7
0.583	3	1.333
1.333	7	3

2. Sum and normalise the rows: (0.615, 0.116, 0.268)
3. Repeat step 1 and 2: (0.614, 0.117, 0.268)
4. Stop if the difference between the priorities of steps 2 and 3 is smaller than the stop criterion

In example 3.3, the value $a_{13} = 7$ is the sum of:

$$a_{13} = a_{11} \cdot a_{13} = 1 \cdot 2 \quad = 2$$

$$a_{13} = a_{12} \cdot a_{23} = 6 \cdot \frac{1}{2} \quad = 3$$

$$a_{13} = a_{13} \cdot a_{33} = 2 \cdot 1 \quad = 2$$

These three lines are the direct and indirect comparisons deduced by the transitivity rule (2.1). Since the matrix is inconsistent, the three estimations are not equal. The power method of squaring the matrix takes the sum of all the three estimations, taking into account direct and indirect estimations. A consistency can be measured with the Consistency Index (C.I.):

(3.2.) $C.I. = \frac{\ddot{e} - n}{n}$ where n is the dimension

λ is the eigenvalue

and the Consistency Ratio (C.R.)

(3.3) $C.R. = \frac{C.I.}{R.I.}$ where $R.I.$ is the Random Consistency Index

Dimension ***n***	1	2	3	4	5	6	7	8	9	10
R.I.	0	0	0,58	0,90	1,12	1,24	1,32	1,41	1,45	1,49

Figure 3.4: Random Consistency Index

The rank reversal problem for scale inversion is the most pertinent criticism of the eigenvalue method [JOHNS79]. The solution of the eigenequation (3.1) gives the right eigenvector $\vec{p}$, which is not necessary the same as the left eigenvector $\vec{p}'$, solution of $\vec{p}'^{\mathbf{T}} \cdot \mathbf{A} = \lambda \cdot \vec{p}'^{\mathbf{T}} \Leftrightarrow \mathbf{A}^{\mathbf{T}} \cdot \vec{p}' = \lambda \cdot \vec{p}'$. The solution depends on the formulation of the problem! This right and left inconsistency (or asymmetry) arises only for inconsistent matrices with a dimension higher than three [SAATY84a].

3.2.3 The Geometric Mean

The priorities are given by the geometric mean (example 3.4).

Example 3.4:
The priorities from the matrix of the example 3.1 calculated with the geometric mean are:

$$p_1 = \sqrt[3]{1 \cdot 6 \cdot 3} = 2.62, \; p_2 = \sqrt[3]{\frac{1}{6} \cdot 1 \cdot \frac{1}{2}} = 0.44, \; p_3 = \sqrt[3]{\frac{1}{3} \cdot 2 \cdot 1} = 0.87$$

Normalizing, we obtain: $\vec{p} = (0.67, 0.11, 0.22)$

The geometric mean minimizes the logarithmic error [CRAWF85]:

$$(3.4) \qquad \sum_{i=1}^{n} \sum_{j=1}^{n} \left(\ln(a_{ij}) - \ln(\frac{p_i}{p_j}) \right)^2$$

where a_{ij} is the comparison between i and j

p_i is the priority of i.

This method is insensitive to an inversion of the scale: the geometric mean of the rows and the columns give the same ranking.

[SAATY90] criticizes this method because he sees no reason to work with a logarithmic scale. He adds that the calculation is made only with a row, i.e. the indirect estimations are not considered [SAATY84a], [SAATY84b].

3.3. Simulations

We generated matrices of different dimensions and inconsistencies in a Monte Carlo simulation and compared the results of the five derivation methods. Our study tries to show under which conditions the rankings are similar and which factors influence on ranking contradictions.

[BUDES86] and [TRIAN90] generate randomly matrices and assign them to an inconsistency group. We do not think this method reflects the process used by decision makers when filling a matrix. We also expect the rate of unusable matrices (totally inconsistent matrices) to be very high.

[GOLAN93] have improved the matrix generation process. Only the first row is randomly selected from a uniform distribution in the interval [1, 9]. The other comparisons a_{ij} are randomly selected from the distribution $a_{ij} \in [\frac{(100-k)\cdot a_{1j}}{100\cdot a_{1i}}, \frac{(100+k)\cdot a_{1j}}{100\cdot a_{1i}}]$, where $j > i > 1$ and $k \in [10, 20, \dots ,90]$. This method expects the errors of the decision maker to be multiplicative: $a_{ij} \cdot e_{ij}$, where e_{ij} is the error factor. Multiplicative errors consider low values to be less perturbation sensitive than high values. For example, if we have an error of $e = 2$, the initial comparison 2 is subject to a shift of $2 = (4 \cdot 2) - 2$ units. For the same error, the comparison 4 is subject to a shift of $4 = (4 \cdot 2) - 4$ units.
This method suffers from other problems: The first row contains discrete values from the interval [1, 9], but the other entries contains continuous values which could even be outside the comparisons scale!

Our simulation approach uses an additive error $a_{ij} + e_{ij}$. Additive errors modify all values of the comparison scale equally. All matrix values come from a discrete interval [1, 9] and we discuss how to deal the extremities of the scale.

3.3.1 Description

The experiment is based on five steps:

1) Building inconsistency groups

For dimensions from three to seven, we build five groups of inconsistent matrices based on the consistency ratios ([0, 0.02[; [0.02, 0.04[; [0.04, 0.06[; [0.06, 0.08[; [0.08, 0.1[). We have chosen consistency ratios smaller than 0.1 (Saaty's limit to accept matrices as "near consistent"). For each group, twenty matrices have been generated, which results in a total of 500 matrices.

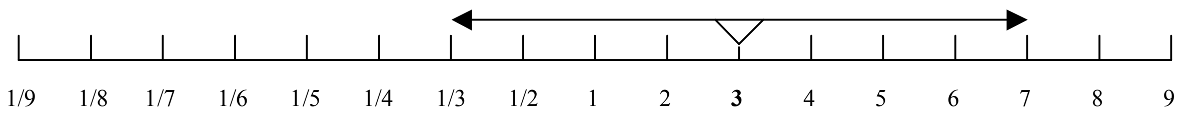
2) Generating a consistent matrix

A consistent matrix is constructed in four steps [ISHIZ04a]. Only the first diagonal is randomly selected from the fundamental scale values [1/9, 1/8, …, 1, 2, …, 9], the other comparisons can be deduced by the transitivity and the reciprocity rules. If a value is induced outside the fundamental scale, the matrix is rejected.

3) Introducing impurities

The number of impurities introduced in the upper part of the matrix is randomly selected from the interval [0, 1, …, $\frac{n^2-n}{2}$], where n is the dimension of the matrix. This impurity is represented as a shift in the comparison scale (equivalent to an additive error term) reaching at most 45 % of the extreme preference 9, i.e ± 4 scale positions (see example 3.5). The comparisons modified by slight perturbations (impurities) are chosen randomly. The process is stochastic; therefore a comparison can be modified by more than one perturbation. The amplitude of the maximal allowed shift is a delicate question. The introduced error should attempt to reflect the inconsistencies of a typical decision maker. It should not be too small to exclude possible scenario. We suppose a maximal shift of 4 units to be a good choice. Anyway, if the impurities are too high and induce strong inconsistencies (i.e. a consistency ratio of more than 0.1) then the matrix is rejected.

Example 3.5:
If the consistent comparison is 3, a shift from at most 4 positions scales is admitted. The new value can be between 1/3 and 7.



The probability of each new value is equal to 100 / 9 = 11.11 %, where the denominator 9 is the number of possible new values. If the original comparison value is greater than 5 or smaller than 1/6 then the number of shift possibilities is less than 9.

Is the decision maker also subject to a border effect and how? Two contradictory theories exist:

- The extreme value is used more frequently than others. Every time the user wishes to enter a value outside the scale, the highest authorised value will be chosen. For example, if a decision maker places a_{ij} = 5 and a_{jk} = 4 then a_{ik} = 9 will probably be entered.
- Psychologist argues that extreme values are less used than middle values (e.g. [BEREK01]).

We suppose that these tendencies compensate for each other and no border effect exists. To avoid a border effect, the probability of each new value remains 11.11% and the original comparison gets the remaining probabilities. For example, if the entry is 8, the new value can be 9, 7, 6, 5, or 4 with a probability of 11.11 % for each number and 8 with a probability of 100 - (5 · 11.11) = 44.45 %.

4) Assigning the matrix to an inconsistency group

The Consistency Ratio (C.R.) is calculated and the matrix is classified into one of the five inconsistency groups (see step 1) or rejected if the C.R. is equal or higher than 0.1.

5) Deriving the priorities

The priorities are calculated with the following five methods:

- Mean of the Normalised Values (MNV)
- Right Eigenvalue Method (REM)
- Left Eigenvalue Method (LEM)
- Geometric Mean of the Rows (GMR)
- Geometric Mean of the Columns (GMC)

The solutions are collected and analysed. The results are shown in the next paragraph.

3.3.2 Results

The five derivation methods do not always result in the same rankings. The number of ranking contradictions with regard to the dimension of the matrix and the consistency ratio is represented in figure 3.5.

	dim 3	**dim 4**	**dim 5**	**dim 6**	**dim 7**	**total**
C.R. < 0.02	0	0	2	2	3	7
C.R. < 0.04	0	2	3	2	6	13
C.R. < 0.06	0	2	4	4	9	19
C.R. < 0.08	0	3	6	10	11	30
C.R. < 0.10	0	4	6	12	16	38
total	0	11	21	30	45	

Figure 3.5: Number of ranking contradictions with regard to the dimension of the matrix and the consistency ratio

We observe that the ranking contradiction phenomenon increases with the dimension of the matrix and the inconsistencies. This is easily explicable. If the number

of alternatives increases, the possibilities of reversal rise too. The process is analogous to the disorder measured by the consistency ratio: when it increases, the probabilities of reversal are higher.
All methods provide the same ranking for matrices of dimension three.

The highest number of ranking contradictions occurs between the right eigenvalue method (REM) and the left eigenvalue method (LEM) (figure 3.6). Relatively few ranking contradictions were found between the right eigenvalue method (REM) and the mean of the normalised values (MNV). It is probably the reason why the mean of the normalised values is widely used as an approximate method of the eigenvalue method.
According to theory, no ranking contradiction appears between the geometric mean of columns (GMC) and rows (GMR).

[%]	**REM**	**LEM**	**GMR**	**GMC**
MNV	34	89	63	63
REM	-		59	59
LEM	-	-	61	61
GMR	-	-	-	0

Figure 3.6: Percent of ranking contradiction between each method

Figure 3.7 represents the mean difference between two reversed priorities. We illustrate the difference between two reversed priorities with the following example 3.6:

Example 3.6:
The priorities are calculated with method A and B: priorities of method A = (0.244; 0.439; 0.11; 0.101; 0.106) and priorities of method B = (0.25; 0.455; 0.104; 0.098; 0.094). A contradiction arises between ranks 4 and 5. The difference between the two reversed priorities is for the method A: 0.106 – 0.101 = **0.005** and for the method B: 0.098 – 0.094 = **0.004**.

This difference seems to be independent of the dimension of the matrix. The highest values can be observed for the left eigenvalue method (LEM). These differences are small (≤ 0.021) when compared to the comparison scale used (interval of 1 unit).

	MNV	REM	LEM	GMR	GMC
Dim 4	0.008 ± 0.000	0.006 ± 0.000	0.022 ± 0.001	0.005 ± 0.000	0.006 ± 0.000
Dim 5	0.008 ± 0.000	0.007 ± 0.000	0.015 ± 0.000	0.008 ± 0.000	0.013 ± 0.000
Dim 6	0.015 ± 0.000	0.018 ± 0.001	0.018 ± 0.001	0.010 ± 0.000	0.012 ± 0.000
Dim 7	0.009 ± 0.000	0.010 ± 0.000	0.012 ± 0.000	0.008 ± 0.000	0.008 ± 0.000

Figure 3.7: Mean difference between two reversed priorities classed by dimension and method

3.4 Conclusion

Five priority derivation methods have been compared in a Monte Carlo simulation. To obtain the same ranking of priorities with the five methods, it is preferable to decrease the dimension and the inconsistencies of a matrix. Anyhow, the differences between the solutions of different methods are minor. Only very close priorities suffer from ranking contradiction. They seem to be due to the uncertainty of subjective judgements. In these cases, a sensitivity analysis can elucidate the decision.

3.5 Summary

- ♦ For the priorities derivation two methods are generally used: the eigenvalue approach and the geometric mean.
- ♦ The eigenvalue method argues that slight perturbations on a consistent matrix induce slight perturbations on the eigenvalue and the correspondent eigenvector.
- ♦ The geometric mean minimises the logarithmic distance.
- ♦ The mean of the normalised values calculates exact priorities for consistent matrices. No theory is known for inconsistent matrices.

- ♦ A simulation based on many inconsistent matrices to study the ranking contradiction phenomena was made. The ranking contradiction is frequent between the right eigenvalue and the left eigenvalue but inexistent between the geometric mean of the rows and the geometric mean of the columns and for matrices of dimension three.

- ♦ The number of ranking contradictions grows with the dimension of the matrix and the consistency ratio.

- ♦ The differences between the solutions of different methods are minor. Only very close priorities suffer from ranking contradiction. In these cases, a sensitivity analysis can elucidate the decision.

Chapter 4

Curriculum

4.1 Introduction

This chapter describes the three families of problems composing the curriculum of AHP-Tutor [ISHIZ03]. The main prerequisite of the exercises is the Weighted Sum Model (see for example [CHEN92, p.36] and [TRIAN00, p. 6-7]). The goal is to illustrate the theoretical basis of AHP (chapter 2 and 3). In particular, the exercises are supposed to …

- prove the eigenvalue approach in a consistent matrix
- recognize the different types of comparisons
- build a consistent or near consistent matrix
- calculate the priorities with:
  - the maximal right eigenvalue approach
  - the maximal left eigenvalue approach
  - the geometric mean of the rows
  - the geometric mean of the columns
  - the mean of the normalised values
- calculate the Consistency Index (C.I.)
- calculate the Consistency Ratio (C.R.)
- observe the effect of perturbations on the priorities
- compare the different methods of priorities derivation.

4.2 Presentation of the Exercises

4.2.1 Exercise 1

The objective of the first exercise is to demonstrate the eigenvalue approach in a consistent matrix [SAATY94 p.48-50, SAATY01]. The exercise contains three steps that lead to the eigenvalue equation:

(4.1) $\mathbf{A} \cdot \mathbf{w} = n \cdot \mathbf{w}$

A is the comparison matrix

w is the eigenvector

n is the dimension of the comparison matrix

a) The wording of an exercise gives the alternatives with their priorities. The student is expected to build the comparison matrix with these priorities. For this purpose, he or she will use the formula:

(4.2) $a_{ij} = \dfrac{p_i}{p_j}$

Example 4.1:
A melon (*1*), a pineapple (*2*) and an orange (*3*) weigh 1kg, 0.25 kg and 0.125 kg respectively. For example, with formula (1) the weight comparison between a melon and a pineapple is $a_{12} = \dfrac{1}{0.25} = 4$. The complete comparison matrix is:

$$\begin{bmatrix} 1 & 4 & 8 \\ 0.25 & 1 & 2 \\ 0.125 & 0.5 & 1 \end{bmatrix}$$

b) In the second step, the student is asked to multiply the comparison matrix with the comparison vector, which reproduces the left part of the eigenvalue equation (2.1).

Example 4.2:

$$\mathbf{A} \cdot \mathbf{w} = \begin{bmatrix} 1 & 4 & 8 \\ 0.25 & 1 & 2 \\ 0.125 & 0.5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0.25 \\ 0.125 \end{bmatrix} = \begin{bmatrix} 3 \\ 0.75 \\ 0.375 \end{bmatrix}$$

c) The third part of the exercise should reproduce the right part of (4.1). The student has to find out that the dimension of the comparison matrix is also a common divisor of the calculated vector.

Example:

$$\mathbf{A} \cdot \mathbf{w} = \begin{bmatrix} 1 & 4 & 8 \\ 0.25 & 1 & 2 \\ 0.125 & 0.5 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0.25 \\ 0.125 \end{bmatrix} = \begin{bmatrix} 3 \\ 0.75 \\ 0.375 \end{bmatrix} = 3 \cdot \begin{bmatrix} 1 \\ 0.25 \\ 0.125 \end{bmatrix} = n \cdot \mathbf{w}$$

4.2.2 Exercise 2

a) Construction of a consistent or near consistent matrix

The objective of the first part of this exercise is to practice the steps shown in the section 2.2.2 to obtain a consistent or near consistent comparison matrix.

b) Priorities derivation

In the second part of the exercise, the student has to calculate the priorities (section 3.2).

c) Consistency measure

For the right and left maximal eigenvalue method, the system proposes to calculate the consistency index (3.2) and the consistency ratio (3.3). For this purpose the student extracts the eigenvalue (4.2) from the eigenvalue equation (3.1). The choice of the line *i* is free. However, depending on the fixed precision in the calculation of the priorities with the power method, slight variations can be observed.

$$(4.2) \qquad \lambda = \frac{\sum_{i=1}^{n} a_{ji} \cdot p_i}{p_j} \qquad \text{where } j \in [1, 2, \ldots, n]$$

4.2.3 Exercise 3

The five proposed methods produce the same priorities for a constant matrix but not when inconsistencies are present. Two theories clash (section 3.1):

- The methods of the maximal eigenvalue are based on the perturbation theory. They affirm that slight perturbations induce only slight perturbations on the priorities and the associated eigenvector. In particular, the maximal eigenvalue is always equal or greater than the dimension of the comparison matrix.

- The geometric mean which minimises the logarithmic distance.

Since arguing in mathematical terms (like in [ISHIZ04c, p. 48-77]) would overwhelm the average user, we let the student experiment with small changes to the comparison matrix.

The student is requested to build a perfectly consistent matrix with the step-by-step method introduced in the section 2.2.2. Next, he or she must introduce perturbations by slightly modifying the comparisons. The student can, in particular, observe the slight variations of the eigenvalue and eigenvector, the illogical rank reversal by the eigenvalue methods, the rank consistency by the geometric mean of the rows and the columns, and the increase of the ranking contradiction with the dimension or the inconsistencies.

To select a method for the derivation of priorities is still an open problem (chapter 3). The goal of this exercise is not to impose a certain method but to give the student an opportunity to form his own opinion.

This exercise, contrary to the other two, is not guided. The student is free to experiment with all comparison changes. He does not need to know rules but is expected to make observations instead.

4.3 Conclusion

This chapter describes the curriculum of AHP-Tutor, which is composed of three exercises. The first is the easiest but most fundamental, since it demonstrates the validity of the maximal eigenvalue for the priorities derivation. From the wording of the exercise to the solution, the resolution path is imposed. That is why we call it a closed exercise.

The second exercise requires the construction of a consistent or near consistent matrix. It is less constraining, because the student has the choice to allot the comparisons values. The calculation of the priorities and the indices of consistency are rather long and difficult. This part of the system functions in "revolver mode". This means that the calculations are realised by the intelligent tutor and their developments can be consulted. The student is free either to solve the problem in parallel with the system and to check its solution, or to accept without going deeper into the suggested solution.

The third exercise represents a variation of the second. The student is free to modify the comparisons and to observe the effects. Contrary to the first exercise, it is open.

These three exercises have different degrees of constraints. The degree depends on the difficulty of the problem modelling and the pedagogy adopted to teach it.

4.4 Summary

- ♦ The first exercise shows the validity of the maximal eigenvalue for a consistent matrix.
- ♦ The second exercise, after construction of a consistent or near consistent matrix, asks to calculate the priorities with five methods.
- ♦ The determination of the consistency index implies the calculation of the maximal eigenvalue.
- ♦ The third exercise observes the effects of inconsistencies on the priorities from the five derivation methods.
- ♦ The degree of freedom a student has in an intelligent tutoring system depends on the difficulty of the problem modelling and the pedagogy adopted to teach it.

Chapter 5

Description of AHP-Tutor

5.1 Introduction

Chapter 4 exposed the exercises. AHP-Tutor was developed in Visual Prolog 5.2 and runs on MS Windows. We describe its architecture and the language used.

5.2 Programming Language

AHP-Tutor focuses, like previous work of our department, on the expert module of intelligent tutoring systems. For the development of the knowledge-base and the explanatory component the logical programming language Prolog [CLOCK87] is adequate. [LUSTI87, p. 152-159] exhaustively describes the advantages and the disadvantages of this language. In particular, Prolog exhibits the following features:

- It is based on a formal theory (logic of the first order), which is relatively close to natural language
- It is non-deterministic, insofar as it often provides all the solutions without having to specify the order and the sequence of the programming instructions to solve a problem. Nevertheless, it also allows a procedural interpretation
- It allows symbolic and numeric processing
- It supports unification (a variable is replaced by a constant, another variable or a functional expression if it does not contain the variable)
- It supports list processing and research by backtracking.

Disadvantages are the difficult procedural structuring of programs and the unreliability due to variables with a no preset size. AHP-Tutor was developed, with the exception of the online help, in Visual Prolog 5.2. This language is a compiled variant of Standard Prolog, which allows to increase execution speed and to generate safer code. Unfortunately, compilation makes meta-programming difficult (section 6.2).

Visual Prolog supports visual programming which makes it easier to design graphical interfaces. It does, however, not reach the quality of the programming environments in Visual Basic or C++, but was sufficient for our work. If the communication interface should be improved, in particular by adding multimedia functionalities, it would be essential to develop an interface in a procedural language and to create the ITS specific modules in Prolog (see [TRAPP00]).

5.3 Architecture

The architecture of AHP-Tutor mirrors the traditional structure of intelligent tutoring systems (figure 5.1):

- Expert Module:
  - solver
  - explanation component
  - supervisor
- Student Module:
  - performance model
  - error model
  - error recorder
- Pedagogical Module
- Communication Module (user interface).

This decomposition must be looked at like a logical organization of the project modules. It does not mean that all these components are autonomous agents.

Pedagogical expertise is distributed in each component: in the expert module the explanations are pedagogically motivated just like the communication with students. The pedagogical module, with the supervisor, controls the course of the exercises.

The cornerstone of any intelligent tutoring system is the expert module. For this reason, the main emphasis was put on this module, the others are rudimentary.

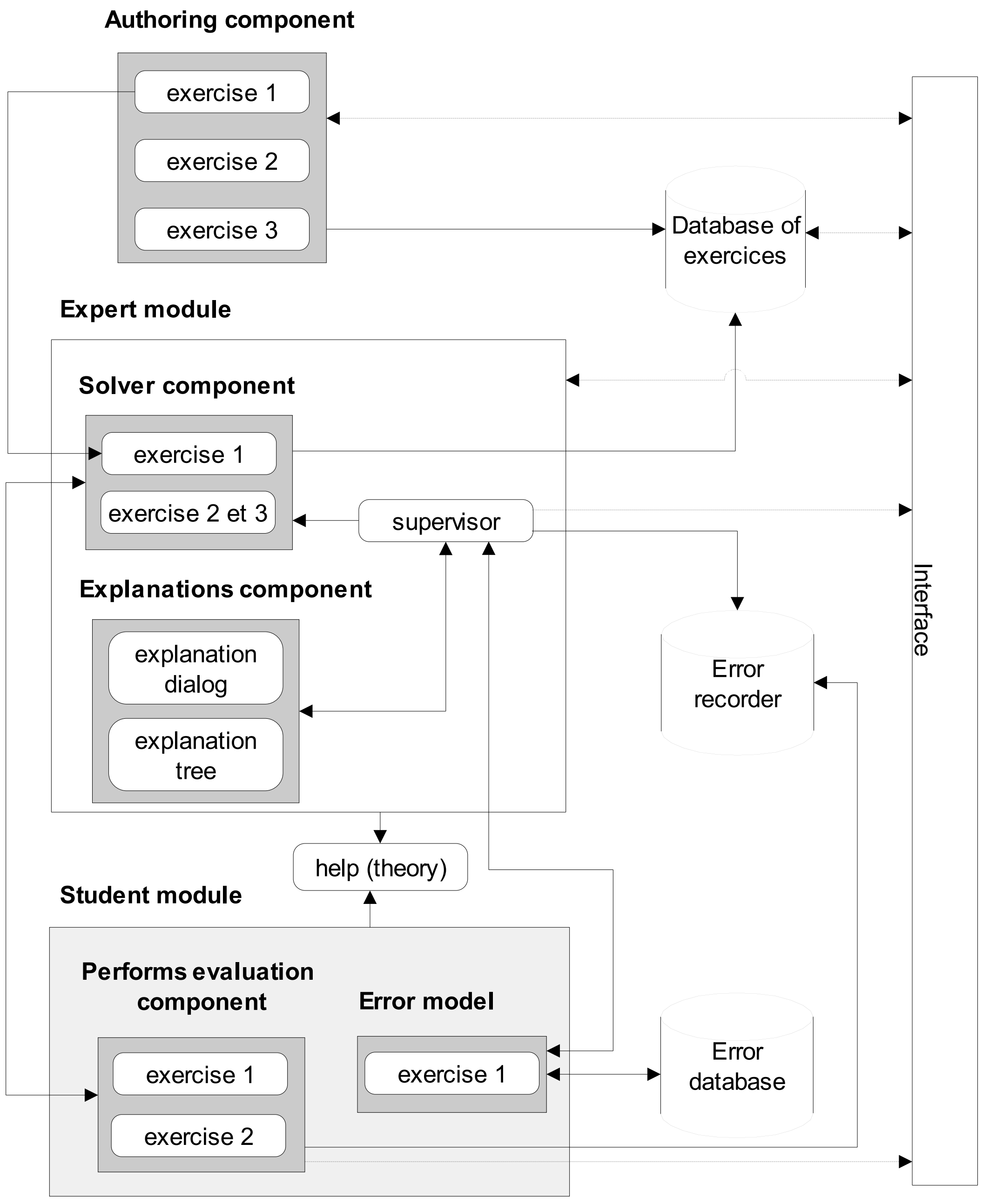


Figure 5.1: Modular architecture of AHP-Tutor. The plain arrows indicate the interactions. The dashed arrows symbolize communications with the student.

A) Authoring component

The authoring component facilitates the formulation of new problems for the teacher. The author is able to view the solution of the first type of exercises.

This function is not possible for the other exercises, because the results depend of the entries of the student.

The author, who is also domain expert, has to specify the number of accepted errors in each part of the exercises, to consider the subjacent theory acquired. This information will be used by the student module.

B) Expert module

The expert module is of a transparent box type. It can solve the problems described in the curriculum (chapter 4) with identical reasoning of a human expert and justify its calculations by the explanation component.

C) Supervisor

The supervisor detects the errors, looks if they are indexed in the error database, announces them to the student, communicates the right answer, proposes an explanation and records the errors.

D) Error model

During tests with students, some errors were recurring. These were based on erroneous rules. We thus decided to count and store them with an adequate explanation.

E) Performs model and error recorder

These two components form the student module based on a performance model. They extract information related to the student's achievement. A database reports the number of errors per rule.

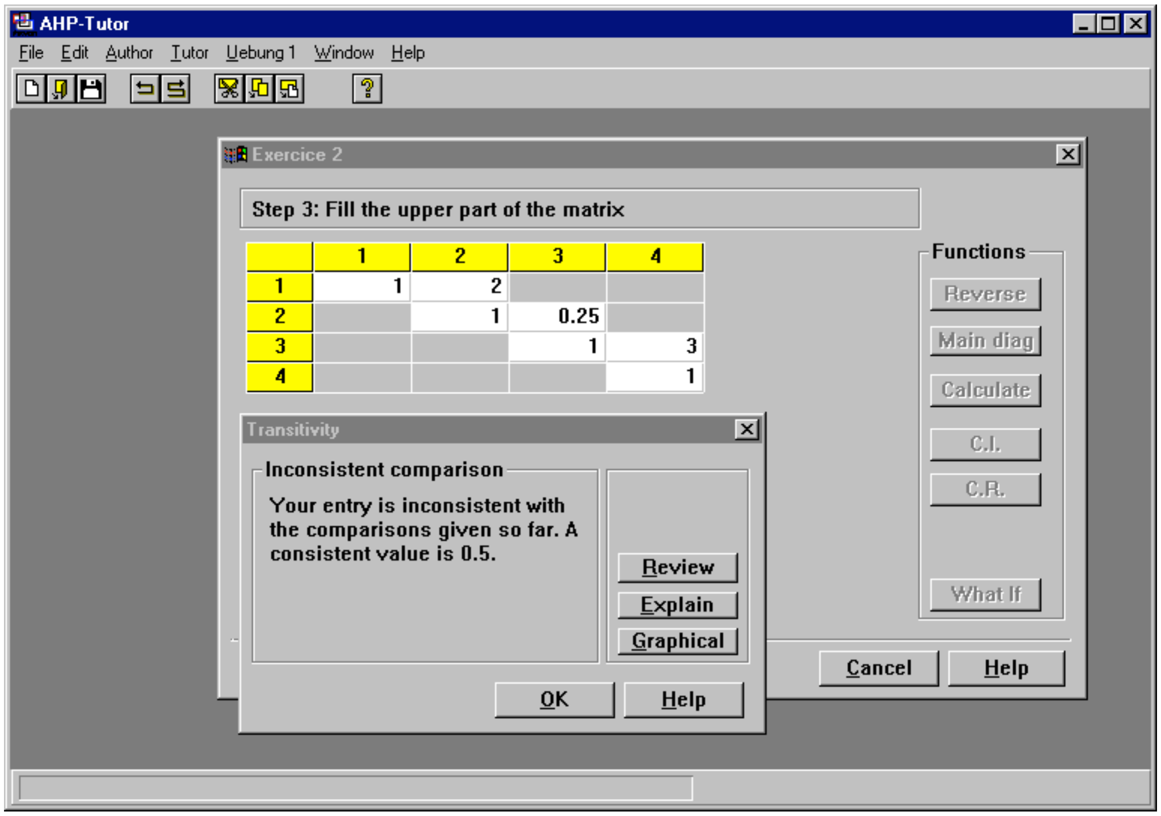
The appraiser decides whether the student acquires a sufficient level to continue the exercise or if it is better to repeat the unity. This judgement is based on the number of errors accepted by the teacher for this exercise (section A: the authoring component).

F) Help

The primary goal of intelligent tutoring systems is to offer exercises to be solved. The referring theory must be acquired differently, for example in class or by books. AHP-Tutor offers a hypertext help containing the necessary theory.

G) Interface

As AHP-Tutor neither aims to reproduce emotional models (via voice recognition, camera,…) nor to propose a highly pedagogical interface (for example by using multimedia or three-dimensional techniques), a simple conventional interface is sufficient. It is window-based and the entries are made by mouse and keyboard (figure 5.2).



Step 3: Fill the upper part of the matrix

	1	2	3	4
1	1	2		
2		1	0.25	
3			1	3
4				1

Inconsistent comparison

Your entry is inconsistent with the comparisons given so far. A consistent value is 0.5.

Figure 5.2: Example of AHP-Tutor interface

5.4 Conclusion

This chapter described the various components of AHP-Tutor. We saw that it follows the classical ITS architecture featuring four interdependent modules. The accent is on a domain-independent explanation component, which is described in the next chapter.

5.5 Summary

♦ AHP-Tutor was implemented in Visual Prolog 5.2. This logical programming language is well adapted to the development of the expert module.

- The architecture of AHP-Tutor is based on four macro-components: the expert module, the student module, the pedagogical module and the interface. However, not all of these components are autonomous agents.

Chapter 6

Explanation Module

6.1 Introduction

Explanations are an integral part of any teaching process. A tutorial system should be able not only to solve problems but also to explain them. For this purpose, the steps leading to the solution must be protocolled. Two techniques are generally used: meta-interpreters and tracers.

6.2 Meta-Interpreter

In meta-programming data are programs [YALCI91]. Since Prolog code and data have the same format (i.e. Horn clauses), this language is particularly appropriate for meta-programming. Meta-interpreters are generally used to add supplementary functions to an existing interpreter.

[RATZ93, p.16-18] and [TRAPP00, p. 49-50] describe simple meta-interpreters protocolling the successive steps to the solution of the problem. Meta-interpreters can be implemented to provide four explanations styles:

- Why is a question being asked (WHY-explanation).
- How has the correct solution been reached (HOW-explanation). When the student asks for a solution, the system solves the problem, protocols the solution path and offers the student a user-friendly way of traversing the protocol. The student can traverse it breadth-first (left to right and right to left) or depth-first (downwards and upwards).
- How would alternative assumptions (e.g. different initial values) modify a solution (WHAT IF-explanation).
- Why is a student answer not correct (WHY NOT-explanation). Supposing the knowledge source is rule-based, the student might then ask why his answer cannot be deduced by the domain and case knowledge. The explanation component would then give the rules which cannot be satisfied.

Meta-programming in standard Prolog is easy because its source code is interpreted and therefore accessible at run-time. Other Prolog systems produce compiled code, which is faster but complicates meta-programming [LUSTI90]. We therefore prefer to implement a tracer [ISHIZ04b].

6.3 Tracer

A tracer collects a detailed protocol (trace) of the execution of a program. To understand how a program reaches its result, we add side effects producing supplementary information on the intermediary states of the execution. This information is analysed and summarised before being presented to the user. The construction of a tracer can be divided into four steps [DUCAS94]:

1. definition of a trace model
2. extraction of information from this model
3. analysis/abstraction of the extracted information
4. visualisation of the result of the analysis.

These four steps are often considered neither explicitly nor separately. In AHP-Tutor [ISHIZ02], for example, the steps 2 and 3 are concomitant.

The tracer of AHP-Tutor reuses the implementation proposed in JA-Tutor [LUSTI87, LUSTI95]. It is adapted to the windowing technology (windows instead of direct output on the screen).

6.3.1 Definition of the Trace Model

An additional argument is added to selected Prolog predicates (figure 6.1). It protocols the execution trace of the predicate. Thus the resolution process and the explanation of the problem are closely connected. A problem cannot be solved without generating a trace and vice versa.

```
Predicate ( Argument1, …, ArgumentN, Trace)
```

Figure 6.1: The additional argument *Trace* is added to selected predicates

The structure of a trace **t** is defined in the explanation component (figure 6.2).

```
Trace = t (predicate, LTrace)   % Structure of trace t
LTrace = Trace*                 % List of trace structures
```

Figure 6.2: The additional trace argument is structured (syntax of Visual Prolog)

The additional trace argument is tree structured (figure 6.3). A trace (or protocol tree) contains rules (intermediate nodes) and facts (terminal nodes or leaves). A fact is a consequence without premise.

$$\textit{Trace} = \mathbf{t}\ (\overbrace{\mathbf{rule}(\textit{Arguments})}^{\text{consequence}},\ \overbrace{[\textit{Trace},\ldots,\textit{Trace}]}^{\text{premise}})\quad \text{\% intermediate node}$$

Trace = **t** (**fact**(*Arguments*), []) % terminal node

Figure 6.3: A trace is a tree

The following example shows the construction of simple trace:

Example 6.1:
(1) knowledge base:

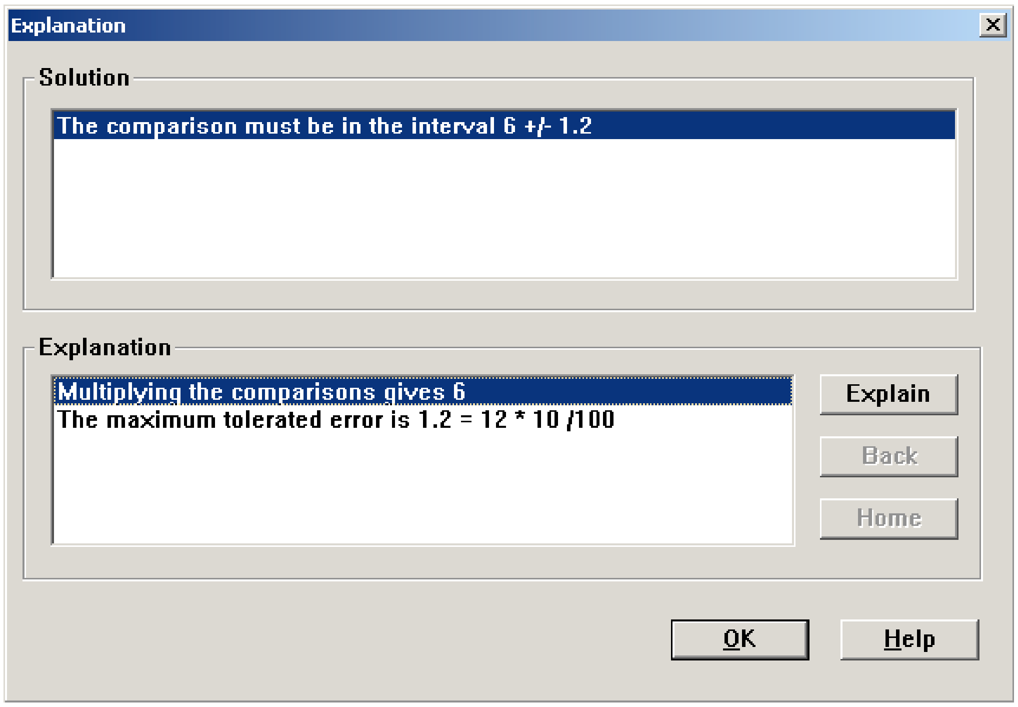
a (**t** (*A*, [*TraceB*, *TraceC*])) :- **b** (*TraceB*), **c** (*TraceC*).
b (**t** (*B*, [*TraceD*, *TraceE*])):- **d** (*TraceD*), **e** (*TraceE*).
c (**t** (*C*, [])).
d (**t** (*D*, [])).
e (**t** (*E*, [])).

(2) The question **a**? instantiates its argument with **t** (*A*, [**t** (*B*, [**t** (*D*, []),**t** (*E*, [])], **t** (*C*, [])]), where *A* is the result and the second argument is the protocol list, which can be read as follows:

A holds because
    B holds because
        D holds because it is a fact
        E holds because it is a fact
    C holds because it is a fact.

6.3.2 Visualisation of Explanations

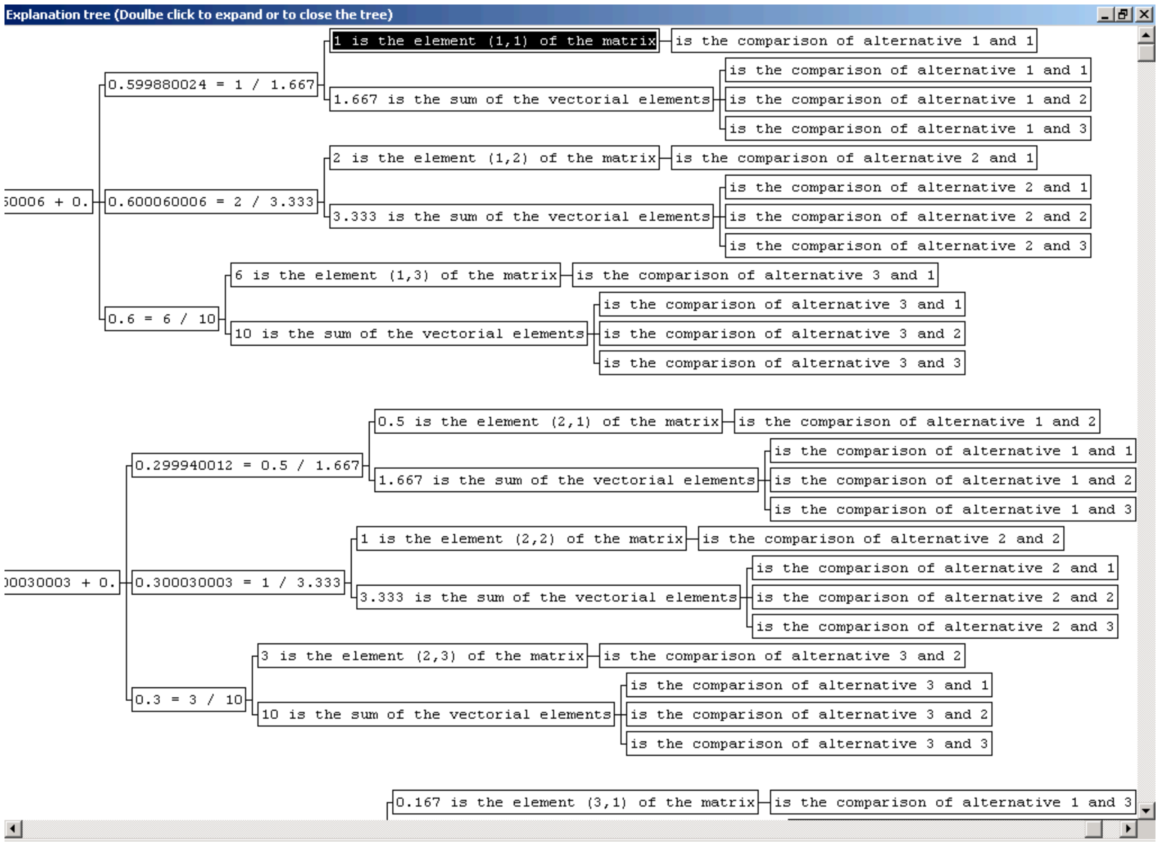
To better understand the implementation, we anticipate the visualisation of explanations. Explanations can be represented textually (dialog of figure 6.4) or graphically (tree of figure 6.5).



The comparison must be in the interval 6 +/- 1.2

Multiplying the comparisons gives 6
The maximum tolerated error is 1.2 = 12 * 10 /100

Figure 6.4: Part of a textual explanation

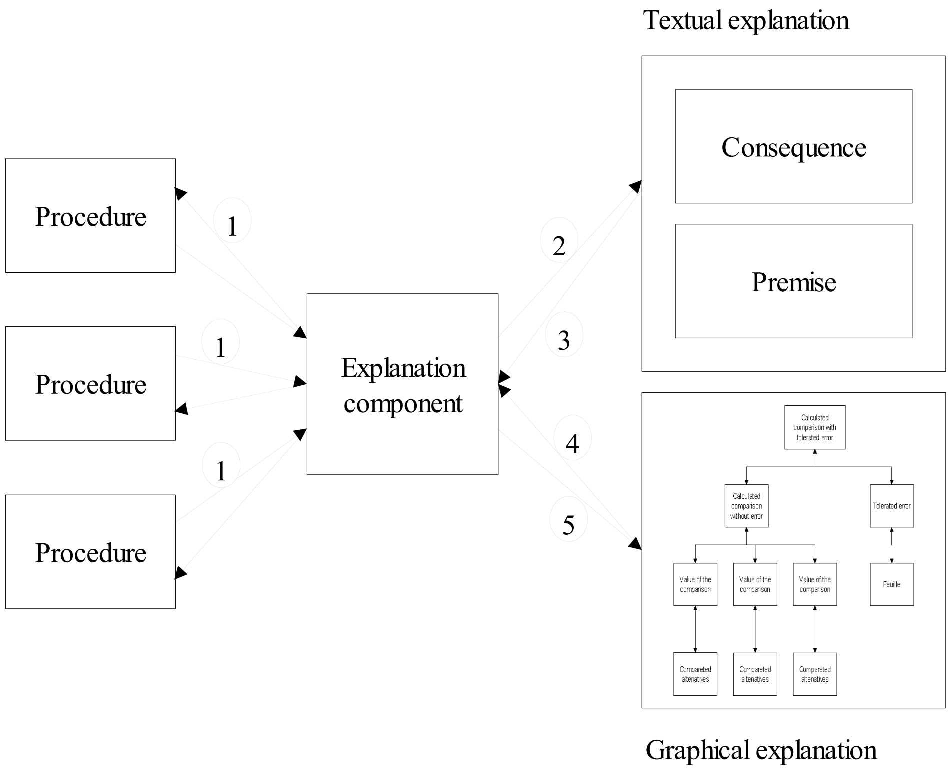

Explanation tree (Doulbe click to expand or to close the tree)

```
50006 + 0.
  0.599880024 = 1 / 1.667
    1 is the element (1,1) of the matrix
      is the comparison of alternative 1 and 1
    1.667 is the sum of the vectorial elements
      is the comparison of alternative 1 and 1
      is the comparison of alternative 1 and 2
      is the comparison of alternative 1 and 3
  0.600060006 = 2 / 3.333
    2 is the element (1,2) of the matrix
      is the comparison of alternative 2 and 1
    3.333 is the sum of the vectorial elements
      is the comparison of alternative 2 and 1
      is the comparison of alternative 2 and 2
      is the comparison of alternative 2 and 3
  0.6 = 6 / 10
    6 is the element (1,3) of the matrix
      is the comparison of alternative 3 and 1
    10 is the sum of the vectorial elements
      is the comparison of alternative 3 and 1
      is the comparison of alternative 3 and 2
      is the comparison of alternative 3 and 3

00030003 + 0.
  0.299940012 = 0.5 / 1.667
    0.5 is the element (2,1) of the matrix
      is the comparison of alternative 1 and 2
    1.667 is the sum of the vectorial elements
      is the comparison of alternative 1 and 1
      is the comparison of alternative 1 and 2
      is the comparison of alternative 1 and 3
  0.300030003 = 1 / 3.333
    1 is the element (2,2) of the matrix
      is the comparison of alternative 2 and 2
    3.333 is the sum of the vectorial elements
      is the comparison of alternative 2 and 1
      is the comparison of alternative 2 and 2
      is the comparison of alternative 2 and 3
  0.3 = 3 / 10
    3 is the element (2,3) of the matrix
      is the comparison of alternative 3 and 2
    10 is the sum of the vectorial elements
      is the comparison of alternative 3 and 1
      is the comparison of alternative 3 and 2
      is the comparison of alternative 3 and 3

    0.167 is the element (3,1) of the matrix
      is the comparison of alternative 1 and 3
```

Figure 6.5: Part of a graphical explanation

The textual explanation displays only one consequence and its conditions (figure 6.4). The user can then navigate freely through the entire tree, gradually looking for deeper explanations. The graphical explanation allows the access to the entire tree by scrolling (figure 6.5). The tree can be expanded or closed by clicking on a node.

When a solution is presented to the student, he or she can ask how it has been achieved. The student can choose its preferred visualisation mode (figure 6.6).



Each arrow corresponds to a call of a new component. The following arguments are passed:

1. a list containing the trace and the displaying mode of the explanation (textual or graphical)
2. a list containing a consequence (head of the list) and its conditions (queue of the list)
3. the answer of the student (closure of the window or navigation in the explanation tree)
4. a list containing the whole explanation tree
5. nothing (closure of the window).

Figure 6.6: Diagram of the analysis and the visualization of the explanations

6.3.3 Information Extraction and Analysis

6.3.3.1 Introduction

The inference engine not only traces rules and facts but also reformulates the trace and comments on a solution. Explicit meta-knowledge in a declarative form is used for this purpose. This paragraph describes how the information contained in the trace is extracted, analysed and commented. The predicate `explain(rule /n)`, for example, adds an explanatory comment to a rule (cf. example 6.2).

Example 6.2
The following clause comments on the result of the transitivity rule:

```
explain (transitivity(Result)) :-
  write("According to the transitivity rule the multiplica-
tion of the comparisons gives " + Result).
```

The extraction of the information from the trace depends on the explanation mode. The next paragraphs detail on the implementation of both textual and graphical explanations.

6.3.3.2 Textual Explanation

The explanations to be displayed are in a list generated by the explanation component. The head contains the consequence and the conditions are in the tail (figure 6.6).

While passing arguments to a dialog in Visual Prolog is easy, returning the arguments without losing the context is not trivial. To keep the context of an explanation dialog we use two global variables. The first global variable records the level in the explanation tree. The following picture denotes the root by the level number 1. After each step downwards the number is incremented by 1 (figure 6.7).

The second global variable communicates one of the following user alternatives to the explanation component:

- Explain one of the conditions (visit a son)
- Return to the predecessor (visit the father)
- Return to the root of the explanation tree
- Leave the explanation dialog.

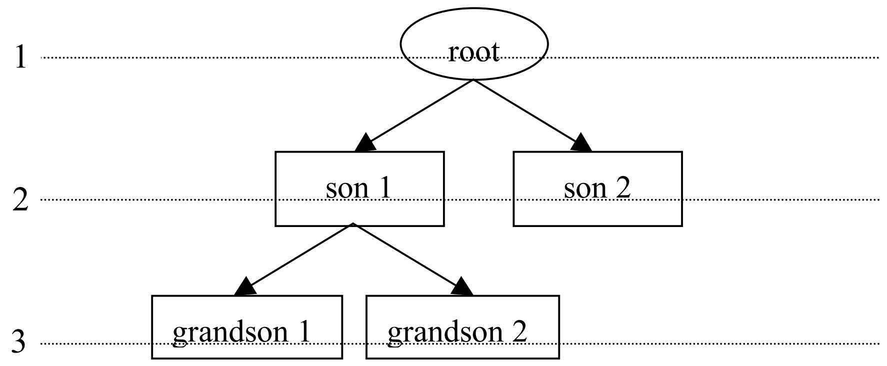


Figure 6.7: Global variable recording the position in the explanation tree

The explanation component is initialised as follows:

- The list of explanations to display is empty
- The initial tree level is 1 (the root)
- The return value is 0.

For each node visited, the comments for the consequence and the conditions are displayed. When a leaf is reached, the system writes the consequence and instead of the lacking premise, it shows the message: "No further explanation possible".

Figure 6.8 analyses the process flow in the explanation component. Backtracking allows to explore alternatives in order to find a valid solution. In figure 6.8, the search for alternatives backtracks to the predicate analysing the student's answer within the explanation dialog (the circle in figure 6.8). Since this predicate is not deterministic, four alternatives are proposed:

1. backtracking (if the returned value is equal or lower than zero)
2. leave the dialog (if the returned value is 1)
3. cut the trace (visit a son) and restart the explanation process
4. restart the explanation process.

The 3rd and the 4th alternatives are unconditional. If the conditions of the first two alternatives do not hold, the 3rd alternative is chosen. The 4th alternative is adopted only when backtracking (figure 6.9). Options 3 and 4 have distinct functions:

- Alternative 3 visits a son. Its tree is extracted and analysed.
- Alternative 4 does not modify the trace. Therefore the displayed explanation is not changed. After having visited a son, the father can be easily reviewed with backtracking.

Backtracking to the root allows reviewing the explanation tree from the start on. The number of backtracking steps is equal to the number of descendants visited.

To know this number, the explanation dialog returns the current level of the tree (figure 6.7). After each backtracking step the tree level is decremented. This process is stopped at the root level.

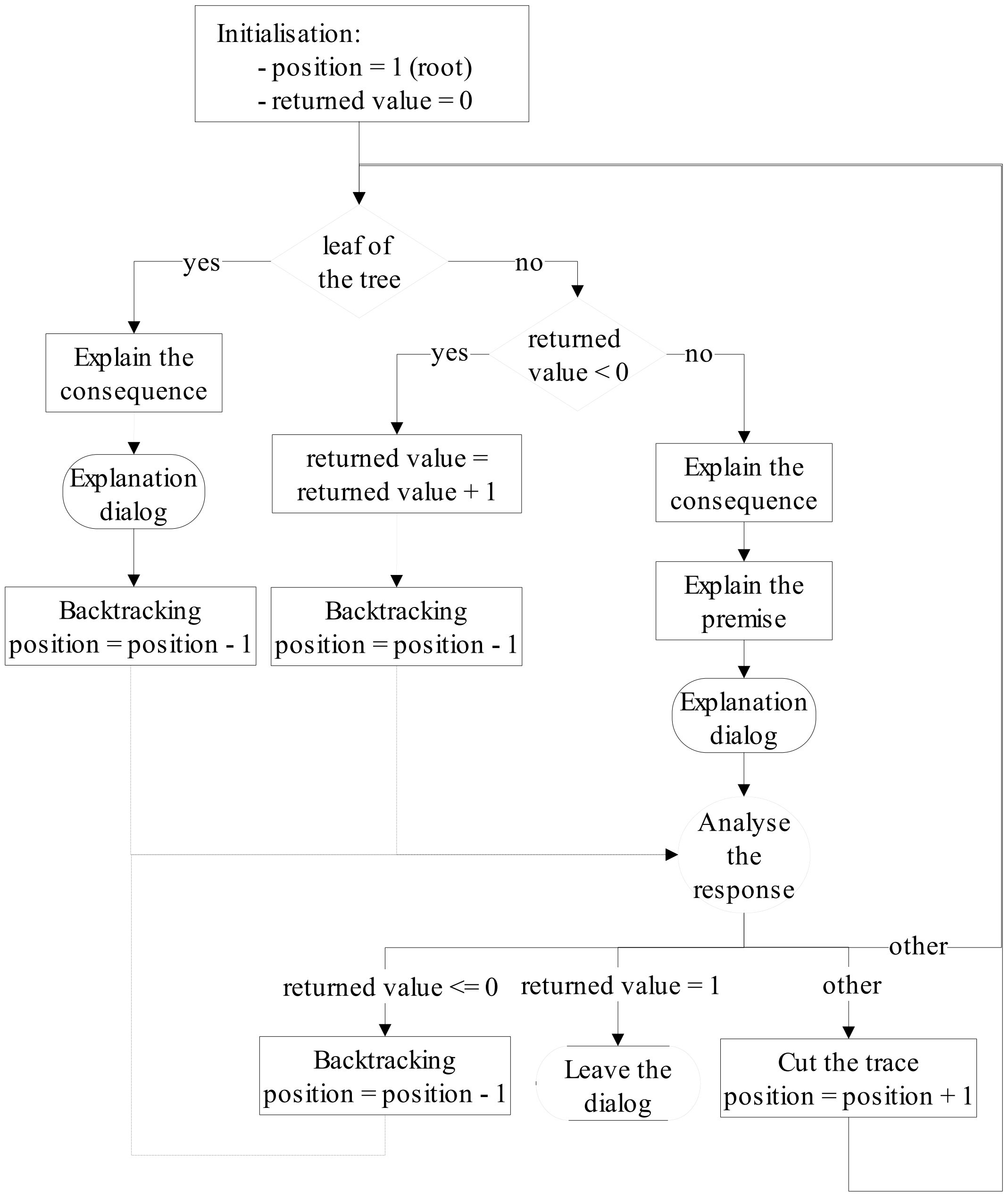


Figure 6.8: Process flow of the explanation component

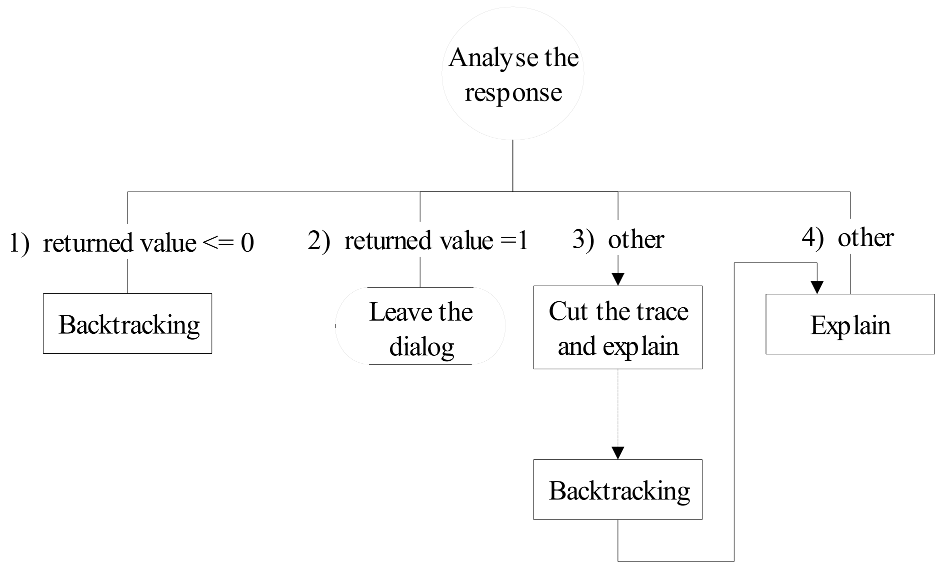


Figure 6.9: Flowchart of a backtracking process. The fourth alternative (explaining the trace without cutting it) is chosen only after backtracking

6.3.3.3 Graphical Explanation

To prepare the generation of graphical explanations, the trace has to be collected by a depth-first traversal of the relevant clauses. Then the comments bound to the rules are read and included in the tree to be displayed (figure 6.10). The tree predicate has the following arguments:

1. the text to be displayed
2. a constant indicating whether the node is initially open or closed (`unmarked /marked`)
3. the sons (conditions) of the node.

```
Tree = tree(<wording of the explanation>, <unmarked/marked>,
  [<Tree of the first condition>, …, <Tree of the n-th con-
dition >])
```

Figure 6.10: Structure of an explanation tree

The figure 6.11 describes the generation of the explanation tree from the trace. The traversal of the trace is an application of depth-first search (figure 6.12). The root is visited then the left subtree and the right subtree are traversed.

```
traverse(Conditions of the virtual root*, Explanations*)  (1)

traverse(Conditions*, Explanations*){                     (2)
  IF Conditions* is empty THEN                            (3)
    Explanations* = empty                                 (4)
  ELSE                                                    (5)
    separate head and tail of Conditions*                 (6)
    separate the Consequence and the Conditions in
    the head of Conditions*                               (7)
    explain (Consequence, Explanation)                    (8)
    traverse(Conditions of the head*, Explanations
    of the descendants*)                                  (9)
    traverse(Tail of Conditions*, Explanations of the
    brothers*)                                           (10)
    Explanations* = [tree(Explanation,marked,
    Explanations of the descendants*)& Explanations
    of the brothers*]                                    (11)
  END IF                                                 (12)
}                                                        (13)
```

Comments:

(1) Extraction of the explanation bound to the root. The arguments of the function are the list of the `Conditions of the virtual root*` (i.e. the trace) and `Explanations*` which contains the returned explanation.
(2) Recursive traversal of the rest of the tree. `Conditions*` is an input argument, `Explanations*` is the returned list.
(8) Explanation of the consequence of the head of `Conditions*`. *Consequence* is an input, `Explanation` is an output argument.
(9) Visit of the first son
(10) Visit of the other sons
(11) Assembling the explanations of the visited node (descendants and their siblings)

Figure 6.11: Algorithm traversing the trace and generating the explanation tree (* means list)

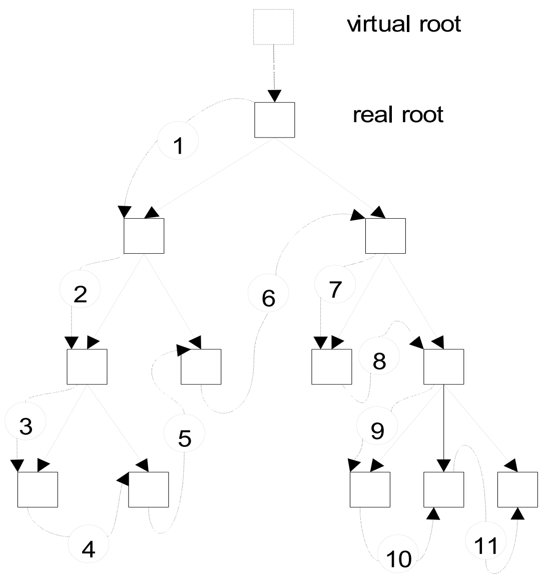


Figure 6.12: Depth-first traversal of a trace

6.4 Conclusion

Explanations are essential in the teaching process. Two implementation architectures can be used: meta-interpreters and tracers. Since meta-interpreters are difficult to implement in a compiled language, we have proposed a tracer which can easily be implemented in a compiled Prolog. The explanation component is modular and independent of the domain. Its reuse in an other context is easy.

We applied our software to students of a course in Decision Support Systems. The practical experience showed that, at least initially, some students had to be motivated to appreciate system provided explanations instead of teacher help. This may be attributed to the prototype frugal user interface. Most students first chose the textual mode, because it allows a better focus on the problematic solution steps. After the clarification of selected steps, the graphical mode was used to get a global view or summary of the resolution process. So both explanation modes complemented each other.

6.5 Summary

♦ For explanation styles can be easily implemented by meta-interpreters:
  - How ?
  - Why ?

- What if ?
- Why not ?

♦ Meta-interpreters and tracers can be used to protocol the steps leading to the solution.

♦ A meta-interpreter can be easily implemented in Standard Prolog. But it is difficult to realise in a compiler like Visual Prolog.

♦ A tracer records the steps leading to the solution of a problem. The recorded protocoll can be extracted, analysed and visualised.

♦ The trace is stored in a supplementary argument added to each Prolog predicate.

♦ An explanation is associated with each used rule. The student can visualise them separately in a verbal dialog or in a graphical tree. The trace and the associated explanations form a recursive tree. A recursive algorithm translates the trace in a displayable tree with the associated explanation.

♦ The explanation dialog allows the student to navigate freely through the entire tree, gradually looking for deeper explanations.

♦ The explanation component of AHP-Tutor is independent of the domain

Chapter 7

Conclusion

We have presented the design and realisation of the intelligent tutoring system AHP-Tutor. This final chapter discusses the results achieved. It points out the goals of the project and briefly presents the framework of the development. It is precisely within this framework where we knew our profitable or unhappy experiments. They will be used for discussions.

The general objective of this work was to help the students of our university, but also any other user of the decision method AHP, to understand how the priorities are derived from a comparison matrix.

In our university [LUSTI02, p.32-43], two methods are taught: the method with the maximal right eigenvalue and the method of the average of the normalised values. Concurrently to these two methods, we also adopted the methods of the geometric mean of the rows, the geometric mean of the columns and the maximal left eigenvalue. The chapter 3.2 reviewed their subjacent theories. They are divided into two groups:

- The perturbation theory, which affirms that the introduction of small impurities in a consistent matrix modifies only slightly the associated priorities and eigenvalues.
- The theory of the distance minimization (logarithmic, in the case of the geometric mean) separating the built matrix and a consistent matrix.

Following this theoretical research, we undertook a study on 500 different inconsistent matrices to observe the divergence of the derived priorities by the various methods. As described in the theory, a ranking contradiction is frequent between the methods of the maximal right eigenvalue and the maximal left eigenvalue. This phenomenon never occurs between the methods of the geometric mean of the rows and the geometric mean of the columns.

We also observed an increase of the ranking contradiction phenomenon with the dimension of the matrix and the consistency ratio. An analysis of the set of reversed priorities pairs showed a weak difference between their two values. In these cases, a clarification by a sensitivity analysis is warranted and if necessary, a statute of parity between the two solutions is more equitable. A use of several methods of priorities derivation is advisable, because it makes it possible to confirm a classification or to underline litigious positions.

After having studied the various methods of derivation of the priorities, we drew up a curriculum comprising three types of exercises with each one representing a different teaching style:

- The demonstration of the method of the maximal eigenvalue uses strict guidance. By this style, we teach the student a presumably optimal way (at least from the point of view of the teacher) leading to the solution. This continuum of strongly structured constraints does not leave any place for the teacher, but simplifies the communication with a student.
- The calculation of the priorities by the various methods functions as a resolver. It makes it possible to use a strong interaction which disburdens the student of calculations (and miscalculations) and allows him to better concentrate on the reasoning.
- The influence of inconsistencies on the priorities and the comparison of the various methods use a learning method by free discovery. As teachers and students employ different approaches of discovery, a role can be reserved for the first: the formulation of problems. He then remains invisible and leaves the student to explore the exercise. Each intervention imposes an external constraint on the student, which has to justify (why this problem it was posed) and to adapt in case of difficulties (to change the wording of the problem).

The second and the third exercises use the same modules, only the initialisation changes. The major difference is the teaching style, which also confers different learning goals.

It seems significant to us to adapt the teaching style not only for the student, but also on the taught subject. It would be interesting to make this process dynamic. However, a dynamic adaptation requires the complicity of a student module which would lay down the next objectives. The present student module of AHP-Tutor is too rudimentary, it does not allow it.

Beside the research project, this project contains also a software development component. AHP-Tutor was entirely developed in Visual Prolog in order to benefit from the properties of an artificial intelligence language. Its power was appreci-

ated in the expert module and in particular in the explanatory component. This compiled version of the standard Prolog allows to accelerate the execution of the program and to make the code safe. This language is, on the other hand, inappropriate to matrix algebra.
The use of a library programmed in C or C++ would be advantageous, however, the construction of an explanatory trace would be more difficult. Visual Prolog does allow only for a limited communicative interface. To improve this aspect, it would be necessary to adopt a client-server architecture. This solution is adopted more and more frequently in computer aided learning. The client, developed in C++ or Java, allows sound, visual animations and sophisticated graphics for an effective communication with the student. The server, implemented in an artificial intelligence language (in general Prolog or LISP), models the expert, the student and the pedagogical components.

Our work focussed on the expert module. It is the cornerstone of the ITSs. AHP-Tutor solves the proposed problems and also offers an explanation by reusing its reasoning and adapting it to the student. The whole reasoning tree is available. However, it becomes quickly incomprehensible for bulky developments.
The major disadvantage of the implemented explanation component is caused by the choice of the technology. It only offers an explanation of the type "how?" and the processes of resolution and explanation are not separable.

The explanation component, readapted from JA-Tutor [LUSTI87], is completely independent from the domain. The other modules (too rudimentary) do not have this property. Furthermore, we cannot affirm that independence of the domain is possible. For example pedagogy is extended to several modules: the choice of the given explanations, the communication with the student, etc… It would be interesting to study in how far a reusable module can be implemented without losing its effectiveness.

AHP-Tutor was used in a beta version by about thirty students during an exercise lesson in our university. If the sample was too weak to draw conclusions on the effectiveness of the system, we made interesting observations. We benefited from criticism by the students to improve certain aspects of our system. In particular the objectives of each exercise are better presented by a user. The students appreciated the explanation faculty of AHP-Tutor. This adds an undeniable advantage over the traditional software of computer based teaching. The students asked further questions than normally. We think that the playful aspect of the lesson encouraged the curiosity of the users and led to a relaxed atmosphere. However, we argue that the teacher must keep a role, because his teaching possibilities remain much more efficient than a machine's, at least at the moment.

Bibliography

[BANA93] Bana E Costa C., Vansnick J.C., *Sur la Quantification des Jugements de Valeurs: l'Approche MACBETH*, Université Paris-Dauphine, Cahier du LAM-SADE N° 117, (1993)

[BARZI87] Barzilai J., Cook W.D., Golany B., *Consistent Weights for Judgments Matrices of the Relative Importance of Alternatives*, Operations Research Letters, Vol. 6 (1), Elsevier Science Publishers, North-Holland, p. 131-134, (1987)

[BARZI90] Barzilai J., Golany B., *Deriving Weights from Pairwise Comparison Matrices: the Additive Case*, Operations Research Letters, Vol. 9, Elsevier Science Publishers, Amsterdam, p. 407-410, (1990)

[BARZI97] Barzilai J., *Deriving Weights from Pairwise Comparison Matrices*, Journal of the Operational Research Society (JORS), Vol. 48 (12), Stockhom, p. 1226-1232, (1997)

[BARZI01] Barzilai J., *Notes on the Analytic Hierarchy Process*, Proceedings of the NSF Design and Manufacturing Research Conference, Tampa, p. 1-6, (2001)

[BEREK01] Berekoven L., Eckert W., Ellenrieder P., *Marktforschung. Methodische Grundlagen und praktische Anwendung*, 9th edition, Gabler Verlag, Wiesbaden, (2001)

[BLANK87] Blankmeyer E., *Approaches to Consistency Adjustments*, Journal of Optimization Theory and Applications, Vol. 45, Kluwer Academic Publishers, Dordrecht [etc…], p. 479-488, (1987)

[BRANS84] Brans J.-P., Mareschal B., Vincke Ph., *PROMETHEE: A New Family of Outranking Methods in Multicriteria Analysis*, in: Brans J.-P. (ed.), Operational Research '84, Elsevier Science Publischers B.V., North-Holland, p. 408-421, (1984)

[BUDES86] Budescu D.V., Zwick R., Rapoport A., *A Comparison of the Eigenvalue Method and the Geometric Mean Procedure for Ratio Scaling*, Applied Psychological Measurement, Vol. 10 (1), p. 69-78, (1986)

[CHEN92] Chen S.-J., Hwang C.L., *Fuzzy Multiple Attribute Decision Making: Methods and Applications*, Lecture Notes in Economics and Mathematical Systems, N° 375, Springer-Verlag, Berlin [etc...], (1992)

[CHU79] Chu A.T.W., Kalabra R.E., Spingarn K.A., *A Comparison of Two Methods for Determining the Weights of Belonging to Fuzzy Sets*, Journal of Optimization Theory and Applications, Vol. 27, Kluwer Academic Publishers, Dordrecht [etc…], p. 531-538, (1979)

[CLOCK87] Clocksin W.F., Mellish C.S., *Programming in Prolog*, Springer Verlag, Berlin [etc…], (1987)

[COGGE85] Cogger K.O., Yu P.L., *Eigenweight Vectors and Least Distance Approximation for Revealed Preference in Pairwise Weight Ratios*, Journal of Optimization Theory and Applications, Vol. 46, Kluwer Academic Publishers, Dordrecht [etc…], p. 483-491, (1985)

[COOK88] Cook W.D., Kress M., *Deriving Weights from Pairwise Comparison Ratio Matrices: An Axiomatic Approach*, European Journal of Operational Research (EJOR), Vol. 37, Elsevier, p. 355-362, (1988)

[CRAWF85] Crawford G., Williams C., *A Note on the Analysis of Subjective Judgement Matrices*, Journal of Mathematical Psychology, Vol. 29, Elsevier Science Publishers, Amsterdam, p. 387-405, (1985)

[DUCAS94] Ducassé M., Noye J., *Logic Programming Environments: Dynamic Program Analysis and Debugging*, The Journal of Logic Programming, Vol. 19-20, Elsevier, p. 351-384, May-June (1994)

[DYER90] Dryer J. S., *Remarks on the Analytic Hierarchy Process*, Management Science, Vol. 36(3), p. 249-258, (1990)

[FISHBU67] Fishburn P.C., *Additive Utilities with Incomplete Product Set: Applications to Priorities and Assignements*, Operations Research Society of America (ORSA), Baltimore, p. 537-542, (1967)

[GOLAN93] Golany B., Kress M*., A Multicriteria Evaluation of the Methods for Obtaining Weights from Ratio-Scale Matrices*, European Journal of Operational Research (EJOR), Vol. 69, Elsevier Science Publishers, Amsterdam, p. 210-202, (1993)

[ISHIZ02] Ishizaka A., Lusti M., *An Intelligent Tutoring System for AHP*, in: Soric K., Hunjak T., Scitovski R. (eds) Proceedings of the ninth International Conference on Operational Research KOI 2002, Grafika d.o.o.,Osijek, p. 215-223, (2002)

[ISHIZ03] Ishizaka A., Lusti M., *Learning how to Derive Priorities from AHP Matrices*, Proceedings of the AIRO2003 – XXXIV Annual Conference of Operational Research Society of Italy, Venice, p.120, (2003)

[ISHIZ04a] Ishizaka A.**,** Lusti M., *An Expert Module to Improve the Consistency of AHP Matrices*, International Transactions in Operational Research (ITOR), Vol.11(1), p. 97-105, (2004)

[ISHIZ04b] Ishizaka A., Lusti M., *A Domain Independent Tracer for Explanations*, ED-MEDIA04 (World Conference on Education Multimedia, Hypermedia & Telecommunication), Lugano, (2004) (accepted)

[ISHIZ04c] Ishizaka A.**,** Développement d'un Système Tutoriel Intelligent pour Apprendre à Dériver les Priorités des Matrices des Comparaisons d'AHP, PhD Thesis, (2004)

[JACQU78] Jacquet-Lagreze E., Siskos J., *Une Méthode de Construction d'une Fonction d'Utilité Additive Explicative d'une Préférence Globale*, Université Paris-Dauphine, Cahier du LAMSADE N°16, (1978)

[JENSE84] Jensen R.E., *An Alternative Scaling Method for Priorities in Hierarchical Structures*, Journal of Mathematical Psychology, Vol. 28, N°3 (September), Elsevier Science Publishers, Amsterdam, p. 317-332, (1984)

[JOHNS79] Johnson C.R., Beine W.B., Wang T.J., *Right-Left Asymmetry in an Eigenvector Ranking Procedure*, Journal of Mathematical Psychology, Vol. 19, Elsevier Science Publishers, Amsterdam, p. 61-64, (1979)

[HARKE97] Harker P.T., *Derivates of Perron Root of a Positive Reciprocal Matrix: With Application to the Analytic Hierarchy Process*, Applied Mathematics and Computation, Vol. 22, Elsevier Science Publishers, Amsterdam, p. 217-232, (1997)

[HWANG81] Hwang C.-L., Yoon K., *Multiple Attribute Decision Making*, Springer Verlag, Berlin [etc…], (1981)

[HINLO83] Hinloopen E., Nijkamp P., Rietveld P., *Qualitative Discrete Multiple Criteria Choice Models in Regional Planning*, Regional Science and Urban Economics, Vol. 13, Elsevier Science Publishers, Amsterdam, p. 77-102, (1983)

[KEENE76] Keeney R.L., Raïffa H., *Decision with Multiple Objectives: Preferences and Value Tradeoffs*, Wiley, New-York, (1976)

[LUSTI87] Lusti M., *Methoden wissensbasierter Systeme bei der Entwicklung von Lernprogramme. Ein Beispiel aus dem betrieblichen Rechnungswesen*, unveröffentlichte Habilitationsschrift, Hochschule St.-Gallen, St.-Gallen, (1987)

[LUSTI90] Lusti M., *Wissenbasierte Systeme*, Bibliographisches Institut / Wissenschaftsverlag, Mannheim [etc...], (1990)

[LUSTI95] Lusti M., *An Authoring Component for Protocol Driven Hypertext Explanations*, in: Greer J. (ed.), Proceedings of Artificial Intelligence in Education 95 (AIED95), Association for the Advancement of Computing in Education, Charlottesville, p. 290-298, (1995)

[LUSTI02] Lusti M., *Data Warehousing und Data Mining*, Springer-Verlag, 2nd edition, Berlin [etc...], (2002)

[MAYST94] Maystre L.Y., Pictet J., Simos J., *Méthodes Multicritères ELECTRE*, Presses Polytechniques et Universitaires Romandes, Lausanne, (1994)

[PAELI76] Paelinck J., *Qualitative Multiple Criteria Analysis, Environmental Protection and Multiregional Development*, Papers of the regional Science Association, Vol. 36, p. 59-74, (1976)

[RÄTZ93] Rätz T., *Erklärungen in Wissenbasierten Lernsystemen am Beispiel eines Tutors zur Normalisierung von Datenbanken*, PhD Thesis, Peter Lang, Frankfurt am Main [etc…], (1993)

[ROUBE79] Roubens M., *Agrégation des Préférences en Présence de Préordres Totaux sur l'Ensemble des Actions et d'une Relation de Type (I ,P,Q) sur les Points de Vue*, 10^{e} réunion du Groupe de travail européen « Aide multicritère à la décision », Liège, (1979)

[ROY68] Roy B., *Classement et Choix en Présence de Points de Vue Multiples (la Méthode ELECTRE)*, Revue Informatique et Recherche Opérationnelle, 2^{e} année, N°8, p. 57-75, (1968)

[SAATY77] Saaty Th. L., *A Scaling Method for Priorities in Hierarchical Structures*, Journal of Mathematical Psychology, Vol. 15, p. 234-281, 1977

[SAATY80] Saaty Th. L., *The Analytic Hierarchy Process, Planning, Priority Setting, Resource Allocation*, McGraw-Hill, United States of America, (1980)

[SAATY84a] Saaty Th. L., Vargas L. G., *Inconsistency and Rank Preservation*, Journal of Mathematical Psychology, Vol. 28, Elsevier Science Publishers, Amsterdam, p. 205-214, (1984)

[SAATY84b] Saaty Th. L., Vargas L. G., *Comparison of Eigenvalue, Logarithmic Least Squares and Least Squares Methods in Estimating Ratios*, Mathematical Modelling, Vol. 5, New York, p. 309-324, (1984)

[SAATY90] Saaty Th. L., *Eigenvector and Logarithmic Least Squares*, European Journal of Operational Research (EJOR), Elsevier Science Publishers, Amsterdam, p. 156-160, (1990)

[SAATY94] Saaty Th. L., *The Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process*, The Analytic Hierarchy Process Series, Vol. VI, RWS Publications, Pittsburgh, (1994)

[SAATY01] Saaty Th. L.*, Decision-making with the AHP: Why is the Principal Eigenvector necessary?*, in: Klaus Dellmann (ed.), Proceedings of the Sixth International Symposium on the Analytic Hierarchy Process (ISAHP 2001), p. 383-396, Bern (2001)

[SAATY03] Saaty Th. L., *Decision-making with the AHP: Why is the Principal Eigenvector necessary?*, European Journal of Operational Research (EJOR), Vol. 145, Elsevier Science Publishers, Amsterdam, p. 85-91, (2003)

[TAKED87] Takeda E., Cooger K.O., Yu P.L., *Estimating Criterion Weights Using Eigenvectors: a Comparative Study*, European Journal of Operational Research (EJOR), Elsevier Science Publishers, Amsterdam, Vol. 29, p. 360-369, (1987)

[TRAPP00] Trapp U., *Komponententechnologien zur Flexibilisierung konventioneller Lernsysteme am Beispiel einer wissenbasierten Erklärungskomponente*, dissertation.de, Berlin, (2000)

[TRIAN90] Triantaphyllou E., Pardalos P.M., Mann S.H., *A Minimization Approach to Membership Evaluation in Fuzzy Sets and Error Analysis*, Journal of Optimization Theory and Applications, Vol. 66, no. 2, p. 275-287, (1990)

[TRIAN00] Triantaphyllou E., *Multi-Criteria Decision Making Methods: A Comparative Study*, Kluwer Academic Publishers, Dordrecht [etc…] (2000)

[YALCI91] Yalcinalp, L.U. *Metaprogramming for Knowledge-Based Systems in Prolog*, PhD Thesis, Case Western Reserve University (1991)

[WEDLE93] Wedley W.C., Schoner B., Tang T.S., *Starting Rules for Incomplete Comparisons in the Analytic Hierarchy Process*, in: Vargas L., Zahedi F. M. (eds), Mathematical and Computer Modelling, Vol. 17, no. 4-5, Analytic Hierarchy Process, Pergamon Press, Oxford, p. 93-100, (1993)

[ZIMME96] Zimmermann H.-J., *Fuzzy Set Theory and its applications*, Kluwer Academic Publishers, Amsterdam (1996)