THE MAUT MACHINE : An Adaptive Recommender System

C. Schmitt, D. Dengler, M. Bauer
DFKI, Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
christian.schmitt@dfki.de, dengler@dfki.de, bauer@dfki.de

Abstract
In this paper we introduce an adaptive recommender system that supports the user in finding interesting entries in an electronic product catalog. The Analytic Hierarchy Process (AHP) lays the foundation for the user preferences. The user interests are specified in terms of desired properties of the ideal product, expressed in the form of constraints (or criteria) on the products basic attributes. Logically related criteria can be combined to form a criteria tree and weights are used to specify the relative importance of these criteria (their contribution impact on the overall rating). An extension of the Multi-Attribute Utility Theory (MAUT) that allows complex and powerful combinations of criteria is used to compute the degree of interest (or utility) of the products regarding the user preferences.
Once the user has defined her preferences she can quickly identify the most promising products in the treemaps display of the catalogue, colour-coding schemes indicating the different degrees of interest and the user having the possibility to sort the catalogue by several of its products basic attributes. The user profile and the catalogue are simultaneously visible and each modification of her preferences is immediately mirrored in the treemaps display that always presents the quality of the catalogue entries w.r.t. the current user preferences. The user can also select a given criteria node in her preferences and see the performance of the catalogue entries for that criteria. This way, rational decision making with multi-criteria objectives is significantly alleviated.

1 Introduction

With the increase of productivity, the diversification and personalisation of goods, companies nowadays offer a large palette of products, making it hard for consumers to find the products that best suit their needs. Even in the modern age of information technology where goods are replaced by information the user is facing an overwhelming amount of information, only a small part of which is really interesting for her. Finding the items that best suit the user's interests is a challenging issue and appears to be as difficult as finding a needle in a haystack, at least for human advisers who can not keep pace with the ever growing amount of goods and information.
For supporting the user while browsing an electronic catalogue we developed an adaptive recommender system, the MAUT machine, that gives a measure of how well the catalogue entries match the user interests. This is the result of our work that focused on the elicitation and representation of multivariate preferences.
MAUT is the abbreviation for Multi-Attribute Utility Theory, a normative method for the evaluation of items involving multiple competing attributes. Relative importance weights are used to specify tradeoffs between the attributes and scaling functions quantify the respective utility (subjective amount of satisfaction) of the attributes. Finally, an aggregation function, typically an additive value function, is used to combine the values from the different attributes. The notion of utility, its suitability for the definition of user preferences and MAUT are introduced in Section 2.
The user not only expects good recommendations but also some reasonable arguments. Such arguments can be produced by a multi-criteria decision method like the Analytic Hierarchy Process (AHP). It consists of a hierarchy of criteria and a list of alternatives that are evaluated against these criteria. Like MAUT tradeoff weights between the criteria are defined and the performance of an item in fulfilling the various criteria can be observed, thus facilitating decision making and reasoning. AHP is described in more details in Section 3.
In Section 4 we describe the evaluation function that is used to evaluate how well a given item match the user interests. The evaluation function is a generalization of MAUT and allows to define the user's interests in a more precise way. In Section 5 we take a look at a concrete example and show how this evaluation function is used to generate recommendations. In Section 6 we introduce Treemaps, a novel visualization technique that displays a tree-like structure in a 2D space with each node represented by a rectangle whose size is determined by the node specific value. Treemaps are used to display the personalized catalogue hierarchy. In Section 7 we briefly describe the technical features of the prototype, before concluding with some directions for future work.

2 Multi-Attribute Utility Theory

Multi-Attribute Utility Theory is a decision-making method used when the decision maker has to take several competing objectives (or requirements) into account. It is a normative/prescriptive method because it tells what we *should* do. Utility quantifies the personal degree of satisfaction of an outcome and in the case of n possible outcomes $X_1,...,X_n$, the expected utility is:

$$EU = \sum_{i=1}^{i=n} p(X_i) \quad U_i(X_i)$$

where $p(X_1),...,P(X_n)$ are the outcomes respective probabilities and $U_1,...,U_n$ their respective utility functions.

While expected utility theory is very good at rating alternatives under uncertainty, the multi-attribute utility theory ([8],[10]) is targeted at tasks that require the rating of items, but without introducing probabilities. Here, the probabilities are replaced by importance weights involving several competing item attributes. The same formula as for the expected utility is used, utility functions are this time used to express the desirability of the attributes. The overall utility for an item with n attributes is defined by the following additive value function:

$$U(X) = \sum_{i=1}^{n} w_i * u_i(x_i) \text{ with } \sum_{1}^{n} w_i = 1 \quad \textit{(1)}$$

where x_i is the value of the *i*th attribute, w_i is the importance weight of the *i*th attribute in comparison to the other attributes and u_i is the utility function of the *i*th attribute.

U is a scoring rule (a special kind of evaluation function) because it assigns a score to a tuple $X = (x_1,...,x_n)$. The most significant improvement of our implementation of MAUT is the ability to use various scoring rules, not only *(1)*. We will delve in details in this later (see Section 4.1).

3 The Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) is a decision-making method developed by Thomas L. Saaty ([5]) in the late 70's. The primary goal of AHP is to find the candidate out of a set of alternatives that best satisfies a set of criteria. The criteria may be divided into sub-criteria and so one, thus forming a hierarchical criteria/decision tree. A leaf criterion is typically the specification of a desired property the candidate should have (in form of an assignment of a candidate's attribute to a desired value) and parent criteria are more abstract criteria that cannot be represented by a single attribute. The criteria need to be prioritized to distinguish between crucial criteria and accessory criteria, so that the achievement of an important criterion will be better rewarded than the achievement of a less relevant criterion. This is usually done by asking the user to do pairwise comparisons between the different criteria (from equal importance of both criteria (score of 1) to absolute importance of one criteria over another (maximal score of 9)) rather than defining directly relative importance weights for the criteria. Relative importance weights are then inferred from pairwise comparisons through some mathematical operations (normalization and computation of eigenvectors) that fall beyond the scope of this paper.

AHP uses MAUT's *(1)* equation to rate the candidates. The utilities indicate how well a candidate fulfils the criteria and the weights are the criteria weights. AHP is therefore a *compensatory decision methodology* because alternatives that are deficient with respect to one or more criteria can compensate by their good performance with respect to other criteria. This is a very desirable feature of AHP because it results in a richer evaluation of the quality of the alternatives.

Let us look at a concrete example. A customer wants to buy a new car and pays special attention to the power aspect of the car, its price and the safety. The price should be inferior to 25000 euro, Airbags and ABS should be available and for the power aspect, it should have a cylinder capacity greater than 2500 and a nominal power greater than 130. A corresponding criteria tree is given in Figure 1.

She already restricted her choice to three candidates: a BMW 325ti, a Mercedes C270 CDI and a Citroën Xsara Picasso 1.8i, whose characteristics are listed in Table 1.

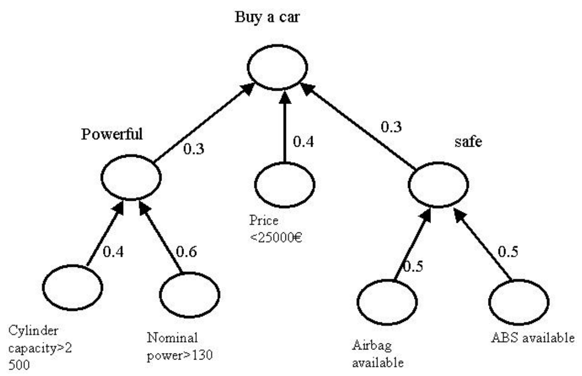


Figure 1: a criteria tree in AHP

Feature\Car	BMW 325ti	Citroën Picasso 1.8i	Mercedes C270 CDI
Airbag	1	1	1
ABS	1	0	1
Price	29000	20000	35000
Cylinder capacity	2495	1749	2685
Nominal power	141	117	125
Fuel	Diesel	Diesel	Petrol
Transmission	Manual	Manual	Manual
Radio	1	1	1
Electric front windows	1	0	1
Power assisted steering	1	1	1
Centralized door locking	0	1	1
Air conditioning	0	1	1

Table 1: features of the candidate cars

To simplify, we will consider the leaf criteria as strict, that is a leaf criterion is either fulfilled or not, resulting in a utility of 0 or 1 (boolean approach). For example, a car with a price of 25001 euros will have a utility of 0 w.r.t. the price criterion (the corresponding utility function is not continuous at 25000). Of course the MAUT Machine uses a fuzzy logic approach through continuous utility. Some of the utility functions used in this example are given in Figure 2.

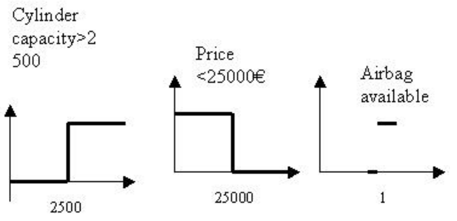


Figure 2: CylinderAbove2500, PriceBelow28000 and AirbagAvail

Now we apply MAUT to the criteria tree in order to calculate the utilities of the non-leaf criteria:

	BMW	Mercedes	Citroen
Cylinder	CylinderAbove2500(2495)=0	CylinderAbove2500(2685)=1	CylinderAbove2500(1749)=0
Nominal Power	NominalPowerAbove130(141)=1	NominalPowerAbove130(125)=0	NominalPowerAbove130(117)=0
Price	PriceBelow25000(29000)=0	PriceBelow25000(35000)=0	PriceBelow25000(20000)=1
Airbag	AirbagAvail(1) =1	AirbagAvail(1) =1	AirbagAvail(1) =1
ABS	ABSAvail(1) =1	ABSAvail(1) =1	ABSAvail(0) =0
Powerful	0.4*0+0.6*1=0.6	0.4*1+0.6*0=0.4	0.4*0+0.6*0=0
Safe	0.5*1+0.5*1=1	0.5*1+0.5*1=1	0.5*1+0.5*0=0.5
Buy a car	0.3*0.6+0.4*0+0.3*1=**0.48**	0.3*0.4+0.4*0+0.3*1=**0.42**	0.3*0+0.4*1+0.3*0.5=**0.55**

Conclusion: Citroën Xsara Picasso 1.8i the best choice because it has the highest utility for criteria "Buy a car" among the three cars. The car matches 55% of the user interest.

In the previous example only *quantitative* criteria were used but AHP also supports *qualitative* criteria like "design" or "cosiness". However, human experts or the user have to rate the alternatives w.r.t. those qualitative criteria because the utility can not be computed for those criteria. The contribution of a criterion to the overall performance of an alternative can then be displayed using bar or pie charts and sensitivity analysis (analysis of the effect of changing criteria weights) decision making.

The entries of an electronic product catalog typically have *basic attributes* like price, brand, availability with numerical, Boolean or string values. While basic products like books have a limited number of attributes, complex products or aggregations of products like computers or cars have thousands of attributes. The number of attributes can be significantly reduced by combining some attributes into a more *abstract attribute*. For example a customer willing to buy a new PC will be more likely interested in the overall CPU performance rather than in the value of each CPU attribute (clock, cache, die size, number of transistors, etc.), especially if she is a novice. This solution saves time to the user as she needs to specify fewer criteria and fastens the different parts of the application (download of the data, computation, display). AHP is the solution of choice for this issue because it allows such a de-composition. The criteria tree, along with the utility functions, the criteria priorities and the scoring rules (see later), represent the user preferences. Representing the profile as a tree is a nice feature because a tree is a well-organized, easy to understand structure. Moreover, parts of the tree can be reused for the definition of other trees, making it easy to build new profiles from predefined profiles. The tedious definition of a tree is therefore significantly alleviated.

4 Scoring rules as evaluation functions

AHP uses the additive value function *(1)* to compute the overall utility of the entries. This way, the hierarchy is not really taken into account because the overall utility can be computed just by summing the contributions of the leafs (weights relative to the sibling criteria have to be converted in weights relative to all criteria), without using the hierarchy in a recursive bottom-up way. The criteria hierarchy contributes basically to the definition of priorities and to the display of intermediate calculations (division of the overall performance into several parts). Since *(1)* is used for every node it is impossible to define nuances about the way criteria should combine. Those nuances are however quite usual and humans often tend to use logical conjunctions like "and" and "or". For example, a user may require both airbags <u>and</u> ABS to be available if she cares a lot for safety while she will require either tennis <u>or</u> table tennis to be available if she wants to play with a racket but does not really care about the concrete option.

4.1 Our solution

Our contribution overcomes these limitations by allowing a different evaluation function to be selected at each node, chosen from a family of functions based on a weighted *scoring rule formula* developed by Fagin and Wimmers ([3]) and *ordered weighted averaging* (OWA) operators which include among other the additive value function and the logical conjunctions "and" and "or". In addition, the computation of the overall utility is now really recursive bottom-up, that is the utility of a node is computed by using its evaluation function and the utilities and priorities of its children, nothing else. The overall utility of an entry can no longer be displayed as a bar or pie diagram. Instead, the criteria tree showing the propagation of intermediate results for the entry can be used to provide insight into the overall utility.

4.1.1 OWA operators

Ordered weighted averaging operators have been introduced by Ronald R. Yager in 1988 ([2]). A summary of OWA operators is given by R. Fullér ([1]). An OWA operator of dimension n has an associated vector A=$(a_1,...,a_n)$ such as $\sum_{1}^{n} a_i = 1$ and a mapping function $\Omega : \Re^n \rightarrow \Re$ defined by :

$$\Omega(x_1,..., x_n) = \sum_{i=1}^{n} a_i \quad x_i = \langle a | x' \rangle$$

where X'=$(x_1',...,x_n')$ is the vector obtained by sorting vector X=$(x_1,...,x_n)$ by decreasing order. That is, x_j' is the *j*th largest element of the bag $\langle x_1,...,x_n \rangle$.

Such an operator is very interesting because it allows to define the Max, Min and Average functions and also functions that are *close* to these fundamental functions, just by slightly changing the associated vector. For example, Max is represented by the vector (1,0,...,0) (only the greatest/first element matters), Min by (0,...,0,1) (only the smallest/last element matters) and Average by (1/n,...,1/n) (each element has the same importance), whereas a Max-like can be represented by (0.8,0.2,0,...,0), in which case the largest value will count for 80%, the second largest for 20% and the other will not be used at all. Since the logical conjunction AND (resp. OR) is equivalent to the Min function (resp. the Max function) (provided that 1 represents true and 0 false), the use of OWA operators provides the user a way to use logical conjunctions with aggregations, like "both airbags <u>AND</u> ABS available" or "tennis <u>OR</u> table tennis available".

4.1.2 Weighted scoring rule

A scoring rule is an assignment of a value to every tuple. In the evaluation of a student, to take the average of the student's grades is a scoring rule. However, in many cases the tuple arguments do not have the same importance (for a computer science student, mathematics skills are more important than sport) and weights are assigned to the arguments. The scoring rule that is typically used to compute the final grade is the weighted *Average* or additive value function:

$$weightedAv\ erage(x_1,...,x_n) = \sum_{i=1}^{n} w_i * x_i \text{ with } \sum_{1}^{n} w_i = 1$$

where x_i is the *i*th grade and w_i its corresponding weight. *WeightedAverage* is a weighted scoring rule that is based on the *unweighted Average*. Does the same applies to other scoring rules ? Is it possible to construct a weighted scoring rule from an unweighted one? Fagin and Wimmers ([3]) were interested in that issue and demonstrated there exists for every unweighted scoring rule a weighted scoring rule based on it. They also indicate how to build it:

Let W=$(w_1,...,w_n)$ be the tuple representing the arguments weights such that $w_1 \geq w_2 \geq ... \geq w_n$ and X=$(x_1,...,x_n)$ the corresponding tuple to be scored. Let f be an unweighted scoring rule. Then there exists a weighted scoring rule F based on f defined by following formula:

$$F(X) = (w_1 - w_2)\ \ f(x_1) + 2\ \ (w_2 - w_3)\ \ f(x_1, x_2) + ... + n\ \ w_n\ \ f(x_1,...,x_n) \quad (2)$$

If the unweighted scoring rule f is the average function $\frac{(x_1 + ... + x_n)}{n}$ then the weighted scoring rule is the weighted average function. Also notice that if weights are equals, then F(X)=f(X).

Hence, the weighted scoring rule formula *(2)* is really a generalization of the "average" case. For the MAUT Machine, we restricted f to the family of OWA operators even if the formula applies to any given f. *(2)* is then applied to every criteria node (each node having its own scoring rule f) where X represents the utilities of the node's children, W their relative weights, and F(X) the node's utility. The parameterization of *(2)* through f is a very nice feature as it allows the MAUT Machine *to use as well the classical MAUT algorithm(f=Mean) as other scoring functions* (Min, Max to simulate And & Or relations).

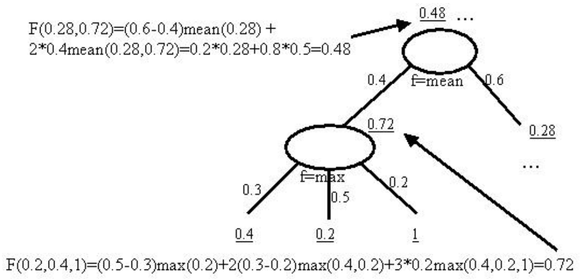

F(0.28,0.72)=(0.6-0.4)mean(0.28) +
2*0.4mean(0.28,0.72)=0.2*0.28+0.8*0.5=0.48
0.48 ...
0.4
f=mean
0.6
0.72
0.28
...
f=max
0.3
0.5
0.2
0.4
0.2
1
F(0.2,0.4,1)=(0.5-0.3)max(0.2)+2(0.3-0.2)max(0.4,0.2)+3*0.2max(0.4,0.2,1)=0.72

Figure 3: The weighted scoring rule in action

To the left is an extract from a criteria tree showing the weighted scoring rule in action. Weights are close to the branches, the tuple arguments are underlined. The bottom node uses f=max, the unweighted score is 1 but the corresponding weight is rather small (0.2), resulting in the weighted score being closer to higher weighted arguments.

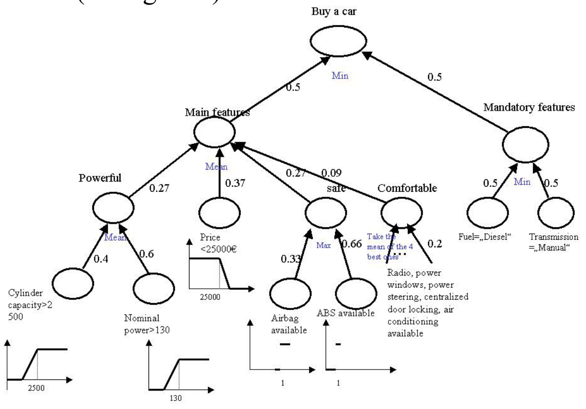
5 Real example

We use the car buying scenario of Figure 1 to compare the classical AHP-MAUT combination with our implementation. First, we apply the MAUT Machine to the criteria tree of Figure 1 (MAUT is used for every node) but took advantage of the fuzzy logic approach used when defining utility functions. For example, the nominal power's utility function "at least 130 hp" will be interpreted as a left linear increasing function, values between 65 and 130 hp getting utilities between 0 and 1.

BMW and Mercedes are similar in terms of performance and security but BMW is cheaper, hence better rated. Citroen is in-between because it lacks on performance but is very cheap, the most important criterion.

Car	Cylinder	Nom.Power	Price	Airbag	ABS	Powerful	Safe	Buy a car
BMW	0.996	1	0.68	1	1	0.4*0.996+0.6*1 =0.998	0.5*1+0.5*1=1	0.3*0.998+0.4*0.68+0.3*1=**0.872**
Mercedes	1	0.923	0.2	1	1	0.4*1+0.6*0.923 =0.954	0.5*1+0.5*1=1	0.3*0.954+0.4*0.2+0.3*1=**0.666**
Citroen	0.399	0.8	1	1	0	0.4*0.399+0.6*0.8=0.64	0.5*1+0.5*0=0.5	0.3*0+0.4*1+0.3*0.5 =**0.742**

Since AHP-MAUT is a compensatory method, it is not possible to exclude (to assign a very bad rating) items that do not fulfill certain conditions. For example, if the user is looking for a diesel, a recommender system based on MAUT may still suggest a petrol car because it is far better than the other cars w.r.t. the other criteria. With the MAUT Machine, mandatory features can be easily defined by using the Min aggregation function and equal weights for mandatory features and other features, as shown in the example below (cf. Figure 4):



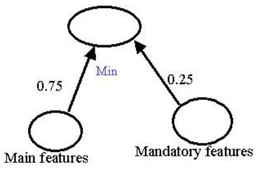
In this example either airbags or ABS have to be available (Max) whereas ABS is twice as important as Airbag, and a new dimension "Comfortable" has been added to the tree to list nice additional features. This dimension uses a general OWA operator that computes the mean of the 4 greatest children utilities, to express that at least 4 of the features should be available. Since Mercedes is a petrol car it will get a rating of 0, whatever the other features are.

Figure 4: profile with different aggregation functions

Among the remaining cars, BMW only has 3 of the 5 "comfortable" features, whereas Citroen has 4 of them, so Citroen is more comfortable than BMW, but the Picasso is missing ABS, the most important "safety" feature, so it will not get the maximum score on safety, even if the Max aggregation function is used. It only scores 2/3, less than if the Picasso have had ABS and no Airbags, in which case it would have scored 1.

	BMW	Mercedes	Citroen
Powerful	0.998	0.954	0.64
Price	0.68	0.2	1
Safe	(2/3-1/3)max(1)+2*1/3*max(1,1)=1	(2/3-1/3)max(1)+ 2*1/3*max(1,1)=1	(2/3-1/3)max(0)+2*1/3*max(0,1)=2/3
Comfortable	(1+1+1+0)/4=0.75	(1+1+1+1)/4=1	(1+1+1+1)/4=1
Main feat.	(0.37-0.27)mean(0.68)+2(0.27-0.27)* mean(…)+3(0.27-0.09) * mean(0.68,0.998,1)+ 4*0.09*mean(0.68,0.998,1,0.75)=0.859	0.692	0.1*mean(1)+0+3*0.18*mean(1,0.64,2/3)+ 0.36mean(1,0.64,2/3,1)=0.813
Mandatory feat.	(0.5-0.5)min(1)+2*0.5min(1,1)=1	(0.5-0.5)min(1) +2*0.5min(1,0)=0	(0.5-0.5)min(1)+2*0.5min(1,1)=1
Buy a car	(0.5-0.5)min(0.859)+2*0.5min(0.859,1)=**0.859**	(0.5-0.5)min(1)+ 2*0.5min(1,0)=**0**	**0.813**

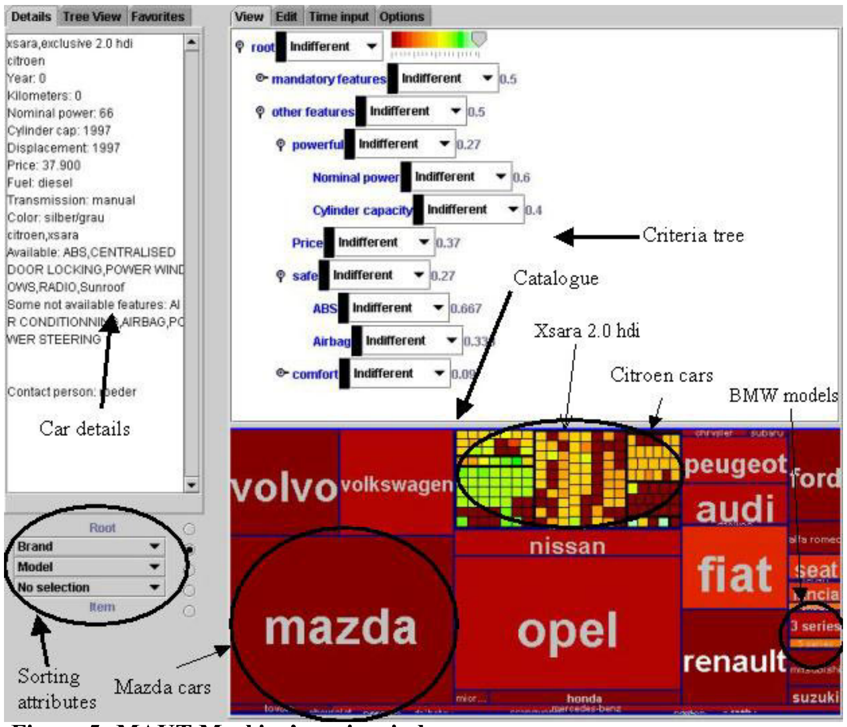
This excluding mechanism is useful in situations where some items should not be taken into consideration, for example when buying a video game a user will not be interested in X-Box games if she has a Playstation. But in other situations mandatory features may be relaxed if an item is excellent at other features, that is if the loss of utility for mandatory features is compensated by the increase of utility for other features. The excluding mechanism can be easily relaxed simply by changing the weights of mandatory features and other features as depicted below:



Here the main features are more important than mandatory features, so the Min function will be softened. Whereas the Mercedes scored 0 in the previous example where both kind of features had the same weight, here the Mercedes scores (0.75-0.25)min(0.692)+2*0.25min(0.692,0)=0.346. If the BMW and the Citroen would have scored 0.1 (instead of about 0.8) then the Mercedes would have been the best choice and the recommender system would have suggested to think about a petrol car because the other cars perform extremely bad on the main features.

6 Treemaps

The Treemap algorithm was developed by Ben Shneiderman from University of Maryland, HCI Lab ([6]). It uses a space filling technique to map a tree structure (e.g. file directory) into nested rectangles with each rectangle representing a node. A rectangular area is first allocated to hold the representation of the tree, and this area is then subdivided into a set of rectangles that represent the top level of the tree. This process continues recursively on the resulting rectangles to represent each lower level of the tree, each level alternating between vertical and horizontal subdivision (slice and dice). The parent-child relationship is indicated by enclosing the child-rectangle by its parent rectangle. That is, all descendents of a node are displayed as rectangles inside its rectangle. Associated with each node is a weight (e.g. size of a directory) and the size of a node‘s rectangle is proportional to that weight. For a complete overview see Ben Shneiderman’s homepage ([9]). Since Shneiderman’s first „slice and dice“ method, many efforts have been made to make the space filling technique more effective in visualizing information hierarchy. The combination of treemaps and AHP have already been suggested by Asahi, Turo and Shneiderman ([7]). They wanted to demonstrate the effectiveness of treemaps for displaying the AHP criteria tree. The user could change the importance of criteria dynamically on the two-dimensional treemap and immediately see the impact on the outcome of the decision. Instead, the MAUT Machine displays the catalogue as a treemap. A catalogue usually has an hierarchical structure with static categories like books, CDs and sub-categories like science-fiction books, novels, jazz CDs, blues CDs, etc. Here the user can dynamically change the catalogue hierarchy by selecting a list of basic attributes the entries will be sorted by, hence a category con-taining entries that share common properties. It is possible to see the data distribution along those attributes. For example it is possible to see how many cars have air conditioning,


xsara,exclusive 2.0 hdi
citroen
Year: 0
Kilometers: 0
Nominal power: 66
Cylinder cap: 1997
Displacement: 1997
Price: 37.900
Fuel: diesel
Transmission: manual
Color: silber/grau
citroen,xsara
Available: ABS,CENTRALISED DOOR LOCKING,POWER WINDOWS,RADIO,Sunroof
Some not available features: AIR CONDITIONNING,AIRBAG,POWER STEERING

Contact person: peder

root Indifferent
mandatory features Indifferent 0.5
other features Indifferent 0.5
powerful Indifferent 0.27
Nominal power Indifferent 0.6
Cylinder capacity Indifferent 0.4
Price Indifferent 0.37
safe Indifferent 0.27
ABS Indifferent 0.667
Airbag Indifferent 0.333
comfort Indifferent 0.09

Root
Brand
Model
No selection
Item

Figure 5: MAUT Machine’s main window.

how many cars among those having the air conditioning also have power windows, etc. You can specify up to three attributes to sort the entries by but there is no limitation from a programming design point of view but in practice too much attributes make the treemap display unreadable.

Imagine a car catalogue where cars are sorted by brand and further by model. The weight associated with each node is the number of cars that are under that node. Leaf nodes represent cars and have a weight of 1.Thus, the categories with the most cars will get more space in the Treemap. Figure 5 shows the MAUT Machine fed with the profile of Figure 4 along with the catalogue where cars have been sorted by brand and model. The treemap displays the final score (“buying a car”) and Citroen cars seem to be the most promising ones.

The user can navigate through the Treemap by changing the depth of sight (the number of underlying levels that are visible) for any given node. For example the depth of sight for Mazda was set to 0 because the Mazda subcategories (Model 626, 323, MX-5 etc.) are hidden whereas BMW’s depth of sight was set to 1 because its models are visible but not the cars. Citroens’s depth of sight equals 2 because its cars are visible. The details of the node under the mouse (the current item) are displayed in a text panel and interesting items can be added to a favorite list. In addition, colors are used to visualize the utilities of the entries and their categories in a more user friendly way. The utility of a category is by default the average of the utilities of its children but the user is free to use another aggregation function based on OWA operators, e.g. by using a max-like aggregation the occurrence of a few high-valued items contained in a Treemap region like BMW could be made visible without performing a further zooming-in. The criteria tree and the catalogue are simultaneously visible and a modification to the criteria tree is automatically reflected in the treemap. That way, the treemap always shows the best candidates w.r.t. the current preferences. When the user clicks on a criterion in the criteria tree the utilities for that criterion are displayed in the treemap. Our treemap implementation is based on the work made by Frederic Vernier on fieldmaps ([4]). Fieldmaps are a little bit different from nested treemaps or squarified treemaps because any element can be optimized to tend to an ideal shape satisfying minimum and maximum height/width ratios. The algorithm uses an evaluation function to compare the possible solutions with the ideal shape and will choose the layout that is the closest to the ideal shape of the most elements. The MAUT Machine uses a restrictive parametrization to ensure nodes are close to squares.

7 The MAUT Machine prototype

The MAUT Machine is based on a client-server architecture: a java applet that consists of a Swing graphical user interface runs in a webbrowser and communicates with a servlet engine interacting with a MySQL database that contains the catalog. The passing of information between the applet and a servlet is done through HTTP tunneling/serialization or XML.

The MAUT Machine has been designed and coded in such a way that it can be deployed as an applet as well as an application without any source code modification. For example, there is no local disc access (forbidden to applets) nor access to system properties, user profiles being stored on the server. The MAUT Machine can be used with any kind of database and for any kind of electronic catalog thanks to technologies like JDBC and the use of XML configuration files that provide an abstract layer essential to the MAUT Machine. Figure 5 shows the MAUT Machine's main panel, consisting of the user preferences at the top right, the treemaps on the bottom right and the details of the entry currently under the mouse on the left (here a Citroen car). Relevant features of the entry are capitalized.

8 Conclusion

We showed how AHP and an extension of MAUT based on a weighted scoring rule and OWA operators could be combined to represent the user preferences in the context of an adaptive recommender system that helps the user to identify interesting items in an electronic catalogue. The hierarchical structure of the preferences and the powerful set of available aggregation functions allow a great freedom for specifying the user interests, resulting in very good recommendations.

The user can dynamically change her preferences and will immediately get updated recommendations in the treemaps display of the catalogue, whose catalogue can be personalized by selecting the attributes to be used for defining catalogue categories. The system also provides clues about the reasons why an item got a good or a bad recommendation, making such a tool a solution of choice for decision-making processes.

The current prototype assumes a good understanding of the theories developed in this paper and we are working on a new, easy to use interface that will require no prerequisite knowledge. Another field of investigation is the data reduction issue that has to be solved for the production version of the MAUT Machine. Indeed, the actual prototype loads the complete database into memory from the server, which is sufficient for a few thousands entries but will not be reliable for bigger databases.

References

[1] R. Fullér, OWA Operators in Decision Making, C.Carlsson ed., *Exploring the Limits of Support Systems*, Åbo Akademis tryckeri, Åbo, TUCS General Publications, No. 3, Åbo, pp. 85-104, 1996.

[2] R.R. Yager, On ordered weighted averaging aggregation operators in multi-criteria decision making, IEEE *Transactions On Systems, Man and Cybernetics* 18(1988), pp. 183-190.

[3] R. Fagin, Edward L. Wimmers, A formula for Incorporating Weights into Scoring Rules, Proceedings of the International Conference on Database Theory, pp. 247-261, 1997.

[4]4 F. Vernier, L. Nigay, Modifiable treemaps containing variable-shaped units, Extended Abstracts of the IEEE Information Visualization, 2000.

[5] T.L. Saaty, The Analytic Hierarchy Process, McGraw-Hill, Inc., 1980.

[6] B. Shneiderman, Tree Visualization with Tree-Maps: 2-d Space-filling Approach, ACM Transactions on Graphics, 11(1), pp. 92-99, 1992.

[7] T. Asahi, D. Turo, B. Shneiderman, Using Treemaps to Visualize the Analytic Hierarchy Process, Information Systems Research 6:4, December 1995, pp. 357-375.

[8] D. von Winterfeldt, W. Edwards, Decision analysis and behavioural research, Cambridge University Press, 1986.

[9] Ben Sheiderman Homepage, http://www.cs.umd.edu/hcil/treemaps/.

[10] W. Edwards, J.R. Newman, Multiattribute Evaluation, Sage Publications, 1982.