UTA - NM: Explaining Stated Preferences with Additive Non-Monotonic Utility Functions

Tomáš Kliegr

Department of Information and Knowledge Engineering, Faculty of Informatics and Statistics, University of Economics, Winston Churchill sq. 4, 130 67 Prague, Czech Republic
tomas.kliegr@vse.cz

Abstract. UTA methods use linear programming techniques for finding additive utility functions that best explain stated preferences. However, most UTA methods including the popular UTA-Star are limited to monotonic preferences. UTA-NM (Non Monotonic) is inspired by UTA Star but allows non-monotonic partial utility functions if they decrease total model error. The shape of the utility functions is determined automatically while overfitting is prevented by balancing the model error with model simplicity. The resulting program is linear and convex, but it requires significantly more CPU time than other UTA methods. The evaluation of the method on a synthetic task achieves the same Pearson Coefficient between the model and stated preferences as Despotis Non-Monotonic UTA. Unlike this method, UTA-NM does not require the provision of information on the shape of the utility function neither is it restricted to one change of shape per criterion.

Keywords: UTA methods, linear programming, preference analysis, utility theory

1 Introduction

In many applications, it is possible to acquire information about preferences of an individual user in the form of a stated order of alternatives (e.g. products) – from the most preferred alternative to the least preferred one. One of the most suitable approaches for processing data of this kind is represented by UTA methods [3, 5], whose output are models explaining user preferences in the form of utility (value) functions. Most UTA methods are not, however, applicable in presence of non-monotonic preferences.

Non-monotonic preferences are not only a theoretical problem known from economics, but also a significant practical issue: consider the notion of the ideal room temperature, one's utility from temperature typically rises up to a certain point after which it again drops [4]. The work presented in this paper studies the problem of non-monotonicity in UTA-methods as represented by the well known UTA-Star [6] method. The monotonicity assumption of the UTA Star is relaxed and the consequential issues with overfitting and normalization tackled. The result of these changes is an experimental method dubbed UTA Non-Monotonic (UTA-NM).

This paper is organized as follows. Section 2 introduces UTA-NM and compares it to the popular UTA-star method. Sections 3 and 4 describe in greater detail the main

differences between the two methods: shape penalization and normalization. Section 5 gives an account of the related research in this field. Section 6 presents an example on which UTA-NM is compared and benchmarked with two other UTA methods. The main contribution of the paper is summarized in Section 7 Conclusions.

2 Description of the UTA-NM

The proposed UTA-NM method is inspired by existing UTA-class methods, specifically by the UTA Star [6] algorithm. UTA methods (from french *UT*ilités *A*ditives) are based on ordinal regression and fall within the framework of multi-criteria decision making. In contrast to many other multi-criteria methods, UTA methods do not expect input from the Decision Maker (DM) in the form of explicitly expressed preferences. The input for the method is constituted by the easier to acquire implicit preferences in the form of a preorder of reference alternatives.

The DM orders the alternatives from the one bringing the highest utility, down to the least preferred one, which is put on the last position. For ordering, the DM uses relations $x > y$ for expressing the preference of alternative x before y and the equivalence relation $x \sim y$ for expressing the indifference of alternatives. The DM does not typically order all alternatives, but only their subset of K reference alternatives A_R; the remaining alternatives are evaluated by the discovered model.

The DM establishes the order of the alternatives based on their description by n criteria $g_1 \dots g_i \dots g_n$. Every criterion is a non-decreasing real function defined on the final set of alternatives A: $g_i: A \rightarrow [g_{i^*}, g_i^*]$, where $g_i(a)$ is a value of alternative a in criterion i and $g(a)$ denotes the vector of values of alternative a on all n criteria.

UTA Star assumes monotonic utility functions: the borders of the image of function g_i constitute the worst (g_{i^*}) and the best (g_i^*) values of the criterion. The more is $g_i(a)$ closer to g_i^*, the better is alternative $a \in A$ evaluated in criterion i. UTA-NM does not assume monotonic utility functions: the worst and the best value of each criterion can be found anywhere in the image of function g_i.

Remaining differences between the methods are a result of relaxing the assumption of monotonicity.

The second important assumption – the additivity of utility functions is preserved in UTA-NM: the utility from alternative a is given by the sum of utilities from the individual criteria $u(a) = \sum_{i=1}^{n} u_i\left(g_i\ (a)\right)$. The utility from criterion i is expressed by partial utility function u_i.

Partial utility functions u_i are piecewise linear; u_i consists of γ_i linear intervals: $[g_i^0, g_i^1]$, …, $[g_i^{\gamma_i - 1}, g_i^{\gamma_i}]$, $g_i^* = g_i^{\gamma_i}$. The domain of u_i is thus $\{g_i^0, g_i^1, .. g_i^j, \dots, g_i^{\gamma_i}\}$. In order to unambiguously determine function u_i it is necessary to find the utilities that correspond to the interval borders, also called breakpoints, g_i^j. These values are determined through variables $w_i^j, j = 0, \dots \gamma_i$, $u_i(g_i^j) = \sum_{k=0}^{j} w_i^k$.

Variables $w_i^j, j = 1, \dots \gamma_i$ represent marginal utilities[1], since they can be computed as a difference of utilities at two consecutive breakpoints $w_i^j = u_i(g_i^j) - u_i(g_i^{j-1})$. UTA-NM allows, in contrast to UTA Star, negative marginal utilities.

The value of partial utility function for *a* is approximated with linear interpolation:

$$u_i(g_i(a)) = u_i(g_i^j) + \frac{g_i(a) - g_i^j}{g_i^{j+1} - g_i^j} \left[u_i(g_i^{j+1}) - u_i(g_i^j)\right] \qquad (1)$$
$$u_i(g_i^j) \leq u_i(g_i(a)) \leq u_i(g_i^{j+1})$$

Values of utility functions are subject to normalization: a) the utility from the worst possible alternative is equal to zero and b) the utility from the best possible alternative, which has the best values in all criteria, is equal to one.

Achieving normalization in UTA Star is simple due to the monotonicity assumption: $u_i(g_i^0) = u_i(g_{i^*}) = w_i^0 = 0, \sum_{i=1}^n u_i\left(g_i^{\gamma_i}\right) = \sum_{i=1}^n u_i\left(g_i^*\right) = 1.$

In the case of UTA-NM the following holds

$$min(u_i(g_i^0), u_i(g_i^1), \dots u_i(g_i^j), \dots, u_i(g_i^{\gamma_i})) = 0 \qquad (2)$$

$$\sum_{i=1}^{n} max(u_i(g_i^0), u_i(g_i^1), \dots u_i(g_i^j), \dots, u_i(g_i^{\gamma_i})) = 1 \qquad (3)$$

The way in which UTA-NM ensures the validity of Equations 2 and 3 using the means of linear programming is described in Section 4.

UTA Star is not always able to find a solution that would be fully compatible[2] with the DM preferences. To find some solution when there is none completely compatible, both UTA Star and UTA-NM add error variables $\sigma^+(a)$, $\sigma^-(a)$ to the partial utility function of alternative a. These variables express the above- and underestimation error of alternative *a*.

$$u'(a) = \sum_{i=1}^{n} u_i\left(g_i\ (a)\right) + \sigma^+(a) - \sigma^-(a) \qquad (4)$$

A nonzero value of $\sigma^+(a)$ or of $\sigma^-(a)$ in the solution indicates that the values of utilities from the alternatives computed from the model need to be adjusted for the errors $\sigma^+(a)$ and $\sigma^-(a)$, so that we get the same order of alternatives as stated by the

[1] Variables $w_i^0, i = 1, \dots n$ are a special case covered to later in this section.

[2] The solution is compatible when the order of alternatives stated by the DM is the same as the order arrived at by sorting the alternatives according to the utilities computed from discovered partial utility functions (the model).

DM. The sum of these errors indicates the quality of the solution found. In UTA, this sum constitutes the whole utility function.

The goal of both UTA and UTA-NM is to find such set of partial utility functions fully determined by values of variables w_i^j that - if used to assign utilities to reference alternatives - would lead to the order of alternatives as close as possible to stated preferences.

Values of variables w_i^j are set based on pair-wise comparisons of two consecutive (in the order stated by DM) alternatives a_k, a_{k+1}. If DM stated that $a_k > a_{k+1}$, then $u'(a_k) - u'(a_{k+1}) > \delta$, and if DM stated that $a_k \sim a_{k+1}$, then $u'(a_k) = u'(a_{k+1})$. Parameter δ is a small positive number that expresses the minimum difference of utilities of two consecutive alternatives.

In UTA-NM, the utility curve is constituted not only by the sum of errors $\sigma^+(a)$ and $\sigma^-(a)$ across all alternatives as in UTA Star, but also by the element *E*, which penalizes the complexity of the discovered model. Without this element UTA-NM would have the tendency to find overfitted solutions. A characteristically trait of an overfitted solution in context of UTA is a high number of changes in the shape of partial utility functions. For the definition of the *change in shape* refer to Section 3.

An outline of the resulting linear program follows:

$$[min]\ z = \sum_{a \in A_R} \sigma^+(a) - \sigma^-(a) + \mathrm{E} \tag{5}$$

s.t.

$$k = 1,2,\dots K-1 \begin{cases} \Delta(a_k, a_{k+1}) > \delta \text{ if } a_k > a_{k+1} \\ \Delta(a_k, a_{k+1}) = 0 \text{ if } a_k \sim a_{k+1} \\ \sigma^+(a_k), \sigma^-(a_k) \geq 0 \end{cases} \tag{6}$$

$$\forall i: min(u_i(g_i^0), u_i(g_i^1), \dots u_i(g_i^j), \dots, u_i(g_i^{\gamma_i})) = 0$$
$$\sum_{i=1}^{n} max(u_i(g_i^0), u_i(g_i^1), \dots u_i(g_i^j), \dots, u_i(g_i^{\gamma_i})) = 1 \tag{7}$$

The principle differences of UTA-NM compared to UTA Star are:

- Admission of negative marginal utilities
- Penalization for shape of partial utility functions
- More sophisticated normalization

Negative marginal utilities were discussed in this section; Section 3 focuses on the penalization element and Section 4 focuses on ensuring the normalization constraints.

3 Shape Penalization

The penalization element E increases the value of the utility function for each point, in which the shape of the partial utility function changes from increasing to decreasing or vice versa (in the following we will refer to this change as a *change of shape*). Change of shape can occur only in the internal points. Since utility functions are piecewise-linear, it suffices to consider the internal breakpoints $g_i^j, j = 1,2, \dots \gamma_i - 1$.

For this purpose, the signs associated with marginal utilities w_i^j, w_i^{j+1} that are adjacent to g_i^j are determined. The sign w_i^j is coded using two binary variables p_i^j and n_i^j: $w_i^j > 0 \rightarrow p_i^j = 1, n_i^j = 0$, $w_i^j < 0 \rightarrow p_i^j = 0, n_i^j = 1$, $w_i^j = 0 \rightarrow p_i^j = 0, n_i^j = 0$. The assignment of values is done through the following linear subprogram:

$$\begin{aligned} w_i^j + \left(1 - p_i^j\right) M &\geq I \\ w_i^j - p_i^j M &\leq 0 \\ w_i^j - \left(1 - n_i^j\right) M &\leq -I \\ w_i^j + n_i^j M &\geq 0 \end{aligned} \qquad (8)$$

Symbol M (so called Big M) is a constant set to a high number, while constant I is set to very small positive number.[3] Constants M and I are used in the linear program for linearization of certain non-linear functions.

The shape of the function between breakpoints g_i^{j-1} and g_i^j can be derived from the sign of w_i^l which is the nearest non-zero marginal utility preceding breakpoint g_i^j.

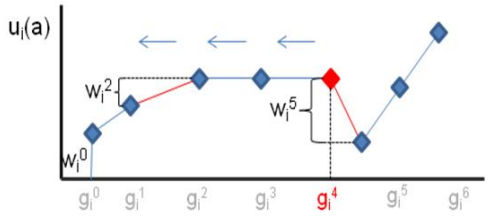
The following subprogram sets the vector of auxiliary binary variables $k_i^j = \{ {}^0k_i^j, {}^1k_i^j, \dots, {}^rk_i^j, \dots, {}^jk_i^j \}$, so that the index r of that variable ${}^rk_i^j$ which is nonzero is equal to the sought index l.

$$q = j, j-1, \dots, 0 \begin{cases} \sum_{r=q}^{j} p_i^r + \sum_{r=q}^{j} n_i^r - \sum_{r=q}^{j} {}^rk_i^j \geq 0 \\ p_i^q + n_i^q - \sum_{r=q}^{j} {}^rk_i^j \leq 0 \end{cases} \qquad (9)$$

$$\sum_{r=0}^{j} {}^rk_i^j \leq 1$$

The process of finding the index l is depicted on Figure 1.

[3] The values used in the example in Section 6 were $M = 1000$ and $I = 0{,}001$.



q	4	3	2	1	0
p_i^r	0	1	1	0	0
n_i^r	0	0	0	0	0
${}^rk_i^j$	0	0	1	0	0
$\sum_{r=q}^{j} {}^rk_i^j$	1	1	1	0	0

Figure 1. The process of setting ${}^rk_i^j$ for index *j=4*. Since the nearest previous nonzero marginal utility at g_i^4 is w_i^2, the value of ${}^2k_i^j$ is set to 1 and index *l* to 2.

Values of signs belonging to w_i^l are saved to binary variables $\tilde{p}_i^j$ and $\tilde{n}_i^j$ with the help of auxiliary binary variables ${}^r\tilde{p}_i^j$, ${}^r\tilde{n}_i^j$:

$$r = 0 \ldots j \begin{cases} \left(1 - {}^rk_i^j\right) M + 2\,{}^r\tilde{p}_i^j - p_i^r \geq 0 & \left(1 - {}^rk_i^j\right) M + 2\,{}^r\tilde{n}_i^j - p_i^r \geq 0 \\ \left(1 - {}^rk_i^j\right) M - {}^r\tilde{p}_i^j + p_i^r \geq 0 & \left(1 - {}^rk_i^j\right) M - {}^r\tilde{n}_i^j + y_i^r \geq 0 \\ {}^rk_i^j - {}^r\tilde{p}_i^j \geq 0 & {}^rk_i^j - {}^r\tilde{n}_i^j \geq 0 \end{cases} \quad (10)$$

$$\tilde{p}_i^j = \sum_{r=0}^{j} {}^r\tilde{p}_i^j \qquad \tilde{n}_i^j = \sum_{r=0}^{j} {}^r\tilde{n}_i^j$$

Variables $\tilde{p}_i^j$ and $\tilde{n}_i^j$ express the shape of the function before the breakpoint g_i^j. These variables code signs with the same combinations of values as variables p_i^j and n_i^j introduced above.

In order to express the shape of the function after the breakpoint g_i^j it suffices to use the sign of w_i^{j+1} that is coded by binary variables p_i^{j+1}, n_i^{j+1}. It is not necessary to introduce sophisticated handling of the situation when $w_i^{j+1} = 0$, since it is clear that in this case the shape does not change.

The change of shape in g_i^j from increasing to decreasing or from decreasing to increasing is characterized by the opposite signs of w_i^l *a* w_i^{j+1}, which are coded by the following two combinations of the auxiliary binary variables holding signs $\tilde{p}_i^j, \tilde{n}_i^j, p_i^{j+1}, y_i^{j+1}$: $\{(1,0,0,1), (0,1,1,0)\}$.

The respective linear subprogram uses two vectors of auxiliary binary variables $x_i^j = \{{}^1x_i^j, {}^2x_i^j, {}^3x_i^j, {}^4x_i^j\}, y_i^j = \{{}^1y_i^j, {}^2y_i^j, {}^3y_i^j, {}^4y_i^j\}$:

$$\begin{aligned} p_i^{j+1} + \tilde{p}_i^j + {}^1x_i^j - {}^1y_i^j = 1 & \qquad {}^1x_i^j + {}^1y_i^j \leq 1 \\ n_i^{j+1} + \tilde{n}_i^j + {}^2x_i^j - {}^2y_i^j = 1 & \qquad {}^2x_i^j + {}^2y_i^j \leq 1 \\ \tilde{p}_i^j + \tilde{n}_i^j + {}^3x_i^j - {}^3y_i^j = 1 & \qquad {}^3x_i^j + {}^3y_i^j \leq 1 \\ p_i^{j+1} + n_i^{j+1} + {}^4x_i^j - {}^4y_i^j = 1 & \qquad {}^4x_i^j + {}^4y_i^j \leq 1 \end{aligned} \quad (11)$$

This subprogram ensures that the vector x_i^j is zero only for the two combinations of values indicating the change of shape. In this case the binary variable e_i^j, which indicates the change of shape in breakpoint g_i^j, is set to 1.

$$\begin{aligned} {}^1x_i^j + {}^2x_i^j + {}^3x_i^j + {}^4x_i^j - M\left(1 - e_i^j\right) &\leq 0 \\ M\,(\,{}^1x_i^j + {}^2x_i^j + {}^3x_i^j + {}^4x_i^j) + M\,e_i^j &\geq 1 \end{aligned} \tag{12}$$

Finally, the penalization element E is constructed as a weighted sum of variables e_i^j across internal breakpoints of all criteria:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{\gamma_i - 1} \mu_i^j \, \delta \, e_i^j \tag{13}$$

The penalization weights for individual interval borders μ_i^j can be set by an expert on the basis of the knowledge of the typical shape of the utility functions. Alternatively, all these weights can be set to the same value $\forall i, j: \mu_i^j = \vartheta$. The value ϑ expresses the preference between the simplicity of the model and its ability to reconstruct the stated preferences on the training set A_R. Generally, $\vartheta < 1$ penalizes the occurrence of a change of shape less than a swapping of two alternatives in the final ranking; value $\vartheta > 1$ then expresses the opposite.

4 Ensuring Normalization

UTA-NM normalizes the maximum and minimum value of utility functions in a similar way as the original UTA method: the utility from the worst criterion value is 0 and the sum of utilities from the best values of all criteria is 1. Hence, the worst possible alternative has utility of 0 and the best possible of 1.

Ensuring these constraints is easy under the assumption of monotonicity. In this subsection, it is shown how this can be achieved in the general case, when the position of the best and the worst criterion value is not known beforehand. UTA-NM applies the following procedure: the value of the global extreme is found for each partial utility function u_i and consequently the normalization constraints are enforced.

Normalization of Maximum Values. The global maximum is sought through comparisons of utilities at breakpoints. First, for each breakpoint g_i^j the highest utility obtained at this and all previous breakpoints $r = 0, \ldots, j$ is found and saved to the variable m_i^j. Variable m_i^j is set through pair-wise comparison of utility $u_i^j = u_i\left(g_i^j\right) = \sum_{r=0}^{j} w_i^r$ and m_i^{j-1}, the highest obtained utility up to the previous breakpoint. Variable m_i^0 is set to utility at the first breakpoint of criterion *i*.

$$m_i^0 = u_i^j$$
$$\text{j} = 1, \dots, \gamma_i : m_i^j = \max\left(m_i^{j-1}, u_i^j\right) \quad \textbf{(14)}$$

The maximum utility at a criterion is then equal to the value of variable $m_i^{\gamma_i}$.

The description of the linear subprogram ensuring this procedure follows. First, the binary variable h_i^j expresses which of the members m_i^{j-1}, u_i^j represents the maximum. If $m_i^{j-1} > u_i^j$, then $h_i^j = 1$, if $m_i^{j-1} < u_i^j$ then $h_i^j = 0$. The value of x_i^j is not important when $m_i^{j-1} = u_i^j$.

For each $j = 1 \dots \gamma_i, i = 1, \dots n$

$$m_i^{j-1} - u_i^j \leq M\, x_i^j$$
$$u_i^j - m_i^{j-1} \leq M(1 - x_i^j) \quad \textbf{(15)}$$

Auxiliary variables a_i^j, b_i^j are set based on the value of x_i^j. For $x_i^j = 1$: $a_i^j = m_i^{j-1}, b_i^j = 0$. For $x_i^j = 0$ is $a_i^j = 0$ and $b_i^j = u_i^j$.

The sought maximum m_i^j is thus given by the sum of values a_i^j and b_i^j. This process is realized by the following linear program:

For each $j = 1 \dots \gamma_i,\ i = 1, \dots n$

$$m_i^{j-1} - a_i^j - M\left(1 - x_i^j\right) \leq 0 \qquad u_i^j - b_i^j - Mx_i^j \leq 0$$
$$a_i^j \geq 0 \qquad b_i^j \geq 0$$
$$a_i^j \leq h_i^j \qquad b_i^j \leq 1 - h_i^j \quad \textbf{(16)}$$
$$a_i^j \leq m_i^{j-1} \qquad b_i^j \leq u_i^j$$
$$m_i^j = a_i^j + b_i^j$$

The maximum value of utility reached at criterion i is thus $m_i^{\gamma_i}$. Once variables $m_i^{\gamma_i}$ are set, ensuring the normalization is straightforward:

$$\sum_{i=1}^{n} m_i^{\gamma_i} = 1 \quad \textbf{(17)}$$

Normalization of Minimum Values. According to the principles of the method set in Section 1, the worst value of each criterion should be assigned the utility of 0. In order to achieve this, it suffices to ensure that the utility at least one breakpoint of each criterion is equal to zero. This can be achieved by first finding out if the utility at breakpoint g_i^j is equal to zero or not and storing the result into the binary variable q_i^j: $q_i^j = 1$ iff $g_i^j = 0$. The following linear subprogram ensures that q_i^j is set:

For each $i = 1, \dots n$

$$j = 0 \dots \gamma_i \begin{cases} u_i^j - \mathrm{M} + \mathrm{M}\, q_i^j \leq 0 \\ u_i^j + q_i^j \geq I \end{cases}$$
$$\sum_{r=1}^{\gamma_i} q_i^r \geq 1 \qquad (18)$$

Once variables q_i^j are set, ensuring the normalization is again straightforward:

$$\sum_{r=1}^{\gamma_i} q_i^r \geq 1 \qquad (19)$$

5 Related Research

There is a large amount of both theoretical research and applications relating to UTA-class methods. However, according to the authoritative survey [3] the only research in the area of dealing with non-monotonicity has been undertaken by Despotis and Zopounidis. Their algorithm [4] (Despotis UTA) is a modification of the UTA Star algorithm. Despotis UTA is based on the assumption that each partial utility function is non-decreasing in the interval g_{i*} to $g_i^{d_i}$ and non-increasing in the interval $g_i^{d_i}$ to g_i^*.

The point d_i at which the function changes its shape is a parameter of the method, which needs to be set for each of the criterions from outside. In contrast to UTA-NM, Despotis UTA is limited to partial utility functions with maximum one change of shape per partial utility function. The position of these points as well as the shape of the functions needs to be known beforehand. Despotis UTA does not ensure the normalization of maximum utility values. The advantage is significantly lower computation complexity compared to UTA-NM. The example described in Section 6 includes a comparison with Despotis UTA.

Although a non-monotonic model may yield a smaller error on the set of reference alternatives, it may suffer from overfitting. For example, in [2] the UTA Star algorithm was applied to a set of individual stated preference data obtained from freight transport managers of Belgian shipping companies, who were interviewed about the importance they gave to six transport attributes: frequency, time, reliability, flexibility, loss, and cost. Two main results were obtained from the analysis: relative weights of the transportation service attributes and partial utility functions for each attribute in the individual shipping companies. The task was performed with the MUSTARD Software [1].

Interestingly, the Kendall coefficient between the ranking of the model and the original ranking didn't reach 1 in any presented individual case, which indicates that it was not possible to fully explain the managers' preference using piece-wise linear monotonic functions. It is likely that less constrained utility functions (e.g. non-monotonic) would lead to better results on A_R concerning Kendall coefficient. However, it is questionable if the underlying managers' preferences were indeed non-

monotonic in some criterion. If it were so, a non-monotonic model would help. Another explanation is that the rankings were inconsistent due to the excessive number of alternatives. In this case, introducing non-monotonicity would lead to overfitting rather than to more accurate models.

6 Example

The goal of the example was to carry out the disaggregation analysis of DM preferences using UTA Star, Despotis UTA and UTA-NM and to compare their results with respect to compatibility with stated and implicit preferences, the shape of the discovered utility curves and time taken to solve the model.

The following case from the domain of dating was selected for the example: woman interested in meeting a new partner (DM) is confronted with the choice of five possible partners, each described in three criteria: g_1: *looks*, g_2 *wittiness*, g_3 *sports*.

Possible values for criteria g_1 and g_2 are {0,1,2,3} and for criterion g_3 {0,1,2,3,4}. All three criteria are for the sake of simplicity ordinal and it can be assumed that generally the higher the value of a criterion the better. Due to the ordinal character of the criteria, the criterion values could also be expressed in a more friendly form e.g. g_3 = {*couch potato*, *walker*, *casual runner*, *frequent runner*, *professional runner*}.

The most important criterion for DM was g_3, where she most preferred medium value of sport endorsement *casual runner*, her second priority was g_1 *looks.* In g_1 and g_2 the preferences were known to be monotonic. This explicit information about the preferences of a DM was not generally accessible to the investigated methods directly, but it was used for creating the stated order of alternatives and for evaluation of the resulting models.

There are five persons - alternatives - between that the DM chooses. Each person is described by a different set of criteria values (Table 1). The hypothetical DM sorted these alternatives in a way reflecting her preferences.

Table 1. Experimental Data

Stated order	Name	Looks	Wittiness	Sport
1.	John	3	1	0
2.	Ashley	1	0	1
3.	Peter	2	2	2
4.	Martin	1	3	3
5.	Stan	1	3	4

All the methods were run with parameters $\gamma_1 = 3, \gamma_2 = 3, \gamma_3 = 4, \delta = 0{,}05$. For Despotis UTA it was necessary to provide an additional parameter $g_3^{d_3} = g_3^3$ locating the position of the point in which the change of shape occurs. The criteria values were sorted from $g_{i=1,2,3^*} = 0$ to $g^*_{i=1,2} = 3, g^*_{i=3} = 4$.

For UTA-NM two tasks were formulated. The Task 1 was regular UTA-NM Task. In Task 2, UTA-NM was constrained by variables w_i^0 set to zero (as in UTA Star). In both UTA-NM tasks, the cost of change of shape was set by $\vartheta = 1$.

The results of individual experiments represented in the form of partial utility functions are presented on Figures 1-4. As seen from Table 2, only UTA-NM produced solutions fully compatible with the stated preferences. The model found by the regular UTA-NM Task 1 was also monotonic[4], but not well reflecting the explicitly stated preferences. The UTA-NM Task 2 with preserved $w_i^0 = 0$ constraint produced a non-monotonic model closely matching explicit preferences.

Table 2. Method comparison

Method	Inferred order of alternatives	Sum of $\sigma^{\pm}$	Pearson coeff.	Changes of shape	Stat. pref.	Time[5] [s]
DM	$a1 > a2 > a3 > a4 > a5$	NA	NA	NA	NA	NA
UTA*	$a1 > a3 = a3 = a4 = a5$	0,15	0,73	0	no	0
Despotis	$a1 > a3 = a3 = a4 > a5$	0,1	0,96	1	yes	0
NM T1	$a1 > a2 > a3 > a4 > a5$	0	1	0	No	40
NM T2	$a1 > a2 > a3 > a4 > a5$	0	1	1	Part.	20

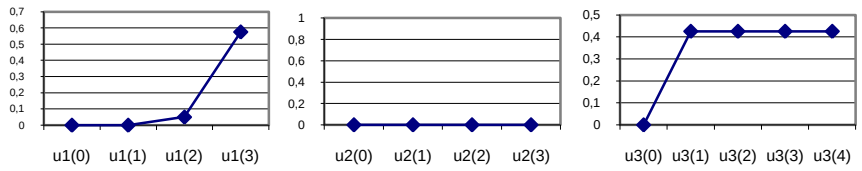


Figure 2. Utility functions found by **UTA Star**

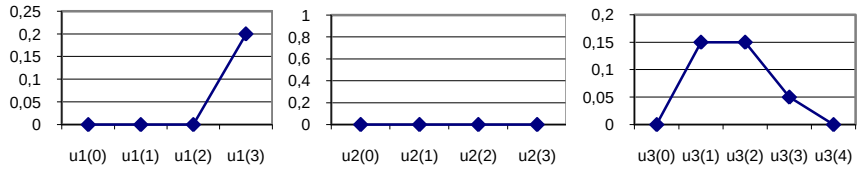


Figure 3. Utility functions found by **Despotis Method**

[4] UTA-NM found a monotonic solution, because UTA-NM was allowed to choose the shape of partial utility functions (by allowing non-zero w_i^0), while UTA-Star was restricted to increasing partial utility functions. Making u_2 and u_3 decreasing was in this case sufficient to find a solution fully compatible with DM preferences – there was no need for UTA-NM to introduce non-monotonicity (for which the objective function would be penalized).

[5] Time for NM T1 is for LPSolve. The Time for NM T2 was achieved in Frontline Solver.

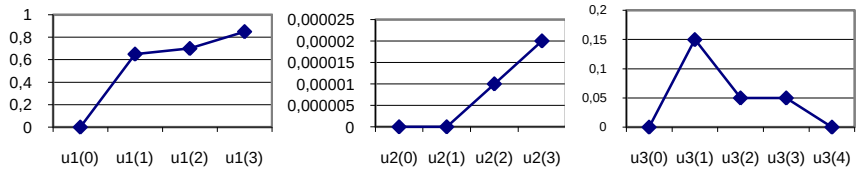


Figure 4. Utility functions found by **UTA-NM with** $w_i^0 = 0$

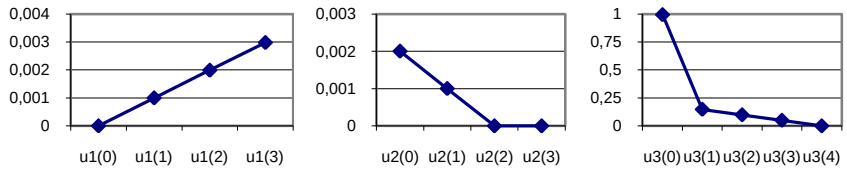


Figure 5. Utility functions found by **UTA-NM**

Performance. The performance of UTA-NM is significantly slower than of other UTA-class algorithms. This can be mainly attributed to the linearizations needed to count the number of breakpoints in which the shape of change occurs and to consequent costly normalization.

The experiments with UTA-Star, Despotis UTA and UTA-NM Task 2 were run in Frontline Solver[6]. UTA-Star and Despotis UTA finished in less than 1 second, UTA-NM Task 2 took 20 seconds to finish. The UTA-NM Task 1 and also UTA-NM Task 2 were also run in LPSolve[7]. UTA-NM Task 1 took 40 seconds to complete. UTA-NM Task 2 did not finish within one hour. The performance of the algorithms can be significantly improved by lifting normalization constraints. When normalization of maximum values was omitted from the linear program, the time to complete UTA-NM Task 1 decreased to 5 seconds. When both normalizations were omitted, the program finished in 2 seconds. The experiments were run on a 32 bit OS on 3 GHz Intel Core2Duo (both solvers utilized only one CPU core).

It should be noted that the performance of UTA-NM, although inferior to other evaluated UTA methods, poses a significant improvement over a baseline scenario, when the number of breakpoints was counted with the help of non-linear functions such as *maximum* or *absolute value* and hence the problem could not be solved by linear programming and non-linear optimization methods had to be applied. In this case, the Frontline GRG Solver and Evolutionary Solver were not able to find an optimal solution even after several hours.

[6] http://www.solver.com
[7] http://sourceforge.net/projects/lpsolve

7 Conclusions

This paper introduced a generalization of UTA method for searching for non-monotonic utility functions with unknown shape. The main focus of the new method is on ensuring the balance between the ability of the model to reconstruct the stated preferences and its complexity. This problem was solved through inclusion of an element penalizing the model complexity into the objective function.

The resulting method, called UTA-NM, keeps, with the exception of the monotonicity, all basic principles of the UTA method. Similarly as UTA, UTA-NM leads to a set of equations that are solvable by linear programming methods. The main contribution of this paper is the expression of the penalization element as a linear programming problem.

The method is available in a form of a Java implementation[8]. The evaluation on a small synthetic dataset indicates that more work remains to be done on performance optimization, since even trivial tasks take tens of seconds to complete.

A possibly fast heuristics inspired by UTA-NM and Despotis-UTA may be based on assuming a maximum number of changes of shape per partial utility function or per all partial utility functions. Once the problem space is limited, it may be feasible to iteratively try all breakpoints in place of $g_i^{d_i}$, changing the partial utility function shape only if it yields a substantial drop in the value of the objective function.

Acknowledgment

This work has been partly supported from grant GACR 201/08/0802 of Czech Grant Agency.

Literature

1. Beuthe, M., Scanella, G.: MUSTARD – User's Guidebook. Le Mons: GTM FUCaM. (2004).
2. Beuthe, M., Bouffioux, Ch., De Maeyer, J., Santamaria, G., Vandresse, M., Vandaele, E. & Witlox, F. "A multi-criteria methodology for stated preferences among freight transport alternatives". In: Reggiani, A. & Schintler, L.A. (Eds.) Methods and Models in Transport and Telecommunications: Cross Atlantic Perspectives. Berlin-Heidelberg-New York, Springer Verlag, pp. 163-179. (2005)
3. Beuthe, M., Bouffioux, C., Krier, C., Michel, M. A comparison of conjoint, multi-criteria, conditional logit and neural network analyses for rank-ordered preference data. MOPGP'06 7th Int. Conf. on Multi-Objective Programming and Goal Programming. Tours, France (2006)
4. Despotis, D., Zopounidis, C.. Building additive utilities in the presence of non-monotonic preferences. Advances in Multicriteria Analysis, pp. 101-114. Kluwer, Dordrecht (1993)
5. Siskos, Y. UTA Methods. Multiple Criteria Decision Analysis: State of the Art suveys, pp. 297-334. Springer, New York (2005)
6. Siskos, Y., Yannacopoulos, D. UTASTAR: An ordinal regression method for building additive value functions. Investigação Operacional 5 (1) , 39–53 (1985)

[8] http://nb.vse.cz/~klit01/uta The program uses the LPSolve open-source library.