On generating utility functions in Stochastic Multicriteria Acceptability Analysis

Luis C. Dias[a], Rudolf Vetschera[b]

[a] *CeBER, Faculty of Economics, University of Coimbra, Portugal*
[b] *University of Vienna, Austria*

Abstract

Stochastic Multicriteria Acceptability Analysis (SMAA) has become a popular Multi-Criteria Decision Aiding tool when some parameters of the decision problem are uncertain or have not been set. SMAA methods use Monte-Carlo simulation of the uncertain values to obtain indicators that inform decision making. While there is considerable literature on sampling of attribute weights, the problem of generating utility values is addressed less frequently. In this paper, we show that direct assignment of random utility values can lead to a biased sample of utility functions. We then introduce new methods to avoid this bias and discuss desirable properties for generating such utility functions. Then, we present a computational study to compare the different methods with regards to the desired properties, concluding that the new approaches perform much better. We also show how these techniques can be extended to consider only utility functions of a specific shape such as concave, convex or s-shaped. Besides SMAA multi-criteria analyses, the techniques studied in this paper can also be used in other contexts, e.g., decision trees and game theory, whenever utility functions need to be simulated to inform decision makers or to study the behavior of different methods.

Keywords: Multiple criteria analysis; Decision Analysis; Utility function; Simulation; Stochastic Multicriteria Acceptability Analysis (SMAA)

1. Introduction

Stochastic Multicriteria Acceptability Analysis (SMAA) (Lahdelma et al., 1998; Lahdelma and Salminen, 2001) is a decision-aiding approach to assess a set of alternatives evaluated in multiple criteria. It allows decision makers to deal with uncertainty concerning preferences, as well as concerning the evaluation of each alternative on each criterion. SMAA methods use Monte-Carlo simulation of the uncertain values to obtain indicators to inform decision making (Tervonen and Lahdelma, 2007; Tervonen and Figueira, 2008). SMAA samples from uncertain performance levels or preference parameters and from this sampling generates a distribution of possible evaluations of alternatives. Even if preferences might be precisely elicited given some time and effort, decision makers can use SMAA to appreciate to what extent they can reach conclusions that are not contingent on such a precise quantification or to direct their information collection effort.

Email addresses: `lmcdias@fe.uc.pt` (Luis C. Dias), `rudolf.vetschera@univie.ac.at` (Rudolf Vetschera)

In the initial versions of SMAA, only attribute weights were considered as unknown parameters, and utilities of outcomes were either assumed to be known, or the (marginal) utility functions were assumed to be linear. More recently, approaches to deal with uncertainty about performance of alternatives or utility functions were added to the SMAA framework in the form of the SMAA-O method (Lahdelma et al., 2003), which uses only ordinal information about outcomes, and the work of Kadziński and Tervonen (2013), who explicitly considered unknown utility functions.

Extending the SMAA framework to uncertain utility functions requires methods to randomly generate utility functions (or utility values) that fulfill certain properties such as monotonicity or a given shape. This is the problem we address in the present paper. We consider a situation in which performance levels of alternatives in each attribute are known, but the utility that a decision maker assigns to these performance levels is unknown. We study methods to assign random utility values to these performance values. As we will show in Section 3, this problem is not trivial. If the distribution of performance levels is skewed, methods that directly assign utility values will introduce a bias towards certain shapes of the utility functions. This paper makes two main contributions: we develop and compare methods to provide an unbiased sample of (monotonic) utility functions of arbitrary shape, and methods to generate an unbiased sample of utility functions of a given shape. To the best of our knowledge, this is the first paper to address the issue of potential biases in randomly generating utility functions and methods to avoid them.

Our contribution is potentially useful in two different areas. The first is decision making with incomplete information about preferences. We complement the SMAA framework with adequate methods to cope with lack of information about utility functions, allowing each criterion to have a utility function of different (given or arbitrary) shape without assuming any functional form. Moreover, the methods proposed here can be used in other contexts that use utility functions, such as decision trees, game theory and negotiation analysis, where simulation approaches can also be used to inform decision making. For instance, Sarabando et al. (2019) proposed the use of simulation to help a mediator propose agreements in bilateral negotiations, with uncertainty about the utilities of the negotiating parties.

The second area for which randomly generated utility functions are useful is that of simulation studies to evaluate decision making methods. Here randomly created utility functions are frequently used to represent the decision maker's "true" preferences. Then a decision model is applied and its results are compared to the "true" preferences. Such studies have a long tradition in evaluating the performance of decision methods using incomplete information (e.g., Moskowitz et al., 1993; Kadziński and Michalski, 2016; Vetschera, 2017), the effects of omitting attributes or other modeling errors in creating a preference model (Stewart, 1996) or the effect of different shapes of utility functions (Lahdelma and Salminen, 2012). These models are typically based on a parametric specification of the (marginal) utility functions, where different functions are created by using random parameters.

Given the variety of contexts where one may wish to randomly generate utilities, we do not distinguish between utility and value functions and use the term utility throughout this paper, even though most applications of SMAA do not involve lotteries and thus do not take risk attitude into account.

In Section 2, we provide a brief overview of related literature. Section 3 shows how direct assignment of utility values may lead to a biased sample of utility functions. In Section 4 we introduce several variants of our general method. A computational study in Section 5 compares these variants. Section 6 extends the method to generate utility functions of a given shape. We specifically focus on generating a concave utility function, but the approach can easily be adapted to convex or s-shaped functions. Section 7 presents our conclusions.

2. Literature review

If a decision maker is not able to specify preferences, SMAA and related methods take the approach of sampling from all preference parameters (weights, utility values etc.) that are compatible with the information the decision maker provides. Originally SMAA considered an additive utility function as the underlying aggregation model (Lahdelma et al., 1998), but extensions of SMAA to other models have also been proposed, (e.g., Hokkanen et al., 1998). SMAA methods have been extensively applied in different domains, as described by Lahdelma and Salminen (2010). Recent applications include work by Cinelli et al. (2017), Corrente et al. (2017), Greco et al. (2018), Kirppu et al. (2018), Li et al. (2018), Loikkanen et al. (2017), Tonin et al. (2017), and Zhu et al. (2017), just to mention a few examples.

SMAA and related methods form one stream of the vast literature on decision making under incomplete information. In their seminal paper, Lahdelma et al. (1998) consider the case of incomplete information on performance levels, where performance is uniformly distributed in a given interval $U(g_{ij} - \Delta_{ij}, g_{ij} + \Delta_{ij})$. They assume a linear utility function and a uniform distribution of attribute weights. The output provided by the first SMAA method consisted in an acceptability index for each alternative, indicating the proportion of the random cases where it was the best alternative (in terms of overall utility). Two other indicators were computed for each alternative: the central weights vector, i.e., the expected center of gravity of the weight vectors that make it the best alternative, and the confidence factor, which is the probability of the alternative being the top one if the central weight vector is chosen (which can be less than 1 due to the uncertain performance values).

SMAA-2 (Lahdelma and Salminen, 2001) improved the original SMAA by introducing the rank acceptability index. The rank acceptability index b_i^r indicates the fraction of cases in which alternative a_i occupies rank r among all alternatives. b_i^1 corresponds to the acceptability index in the original method. Later, Lahdelma et al. (2003) proposed SMAA-O, which assumed only ordinal information about performance. SMAA-O generates random numbers for each attribute and sorts them to match the order of alternatives. Details about these methods and possible implementations can be found in, e.g., Tervonen and Lahdelma (2007); Tervonen and Figueira (2008). More recently, several methods were developed to obtain complete rankings of alternatives from the stochastic information generated in SMAA (Kadziński and Michalski, 2016; Vetschera, 2017).

Approaches to decision making under incomplete information typically refrain from using a parameterized analytical specification of a utility function. Already very early methods for decision making under incomplete information like the UTA method of Jacquet-Lagrèze and Siskos (1982) directly assign utility values to outcomes rather than transforming outcomes via a parameterized utility function. These methods do not involve sampling, but rather estimate one specific function from the preference information provided by the decision maker (e.g., Kadziński et al., 2012).

Only few authors have considered methods based on a direct sampling of utility values. In their paper on integrating SMAA and ROR methods, Kadziński and Tervonen (2013) developed a SMAA approach for a case in which the performance of the alternatives is known, but the utility function is unknown. They generate utility values by sorting random values drawn from a uniform distribution, making no assumptions about properties of the utility function except monotonicity. The same paper proposes the concept of pair-wise outranking indices (and pair-wise winning indices) as outputs that indicate, for each pair of alternatives (a_i, a_j), the proportion of cases for which a_i is not worse than a_j (and for which a_i is better than a_j, respectively). These methods provide a fast and simple way to generate arbitrary (monotonic) utility functions. However, the exact properties of the functions generated by these approaches have not been studied systematically. As we will show in Section 3, they can bias the resulting utility function towards a certain shape, which might distort conclusions.

Furthermore, in some situations it might be necessary to generate utility functions of a specific shape, e.g., a concave utility function if the decision maker is known to be risk averse (or if one wants to specifically study the performance of a decision making method for risk averse decision makers). Of course, it is possible to generate arbitrary monotonic functions and utilize only those functions that have the desired properties, but that approach might require considerable computational effort. Our goal for the present paper is on the one hand to develop methods that will generate an unbiased sample of utility functions of different shapes if no shape is a priori specified, and on the other hand to develop efficient methods that are guaranteed to generate only functions of a desired shape.

3. Direct generation of utility values

Consider the problem of generating random utilities for a set of alternatives $A = \{a_1, ...a_m\}$ on a given criterion (attribute) g, when the performance levels of these alternatives, $g(a_1)$,...,$g(a_m)$, are known. Let $x = (x_0, ..., x_{n(g,A)})$ denote a sorted vector of the possible performance levels in criterion g, i.e., x_0 is the lowest and $x_{n(g,A)}$ is the highest performance value. The number of performance levels can be smaller than the number of alternatives if some alternatives have the same value. To simplify notation, we will refer to $n(g, A)$ as n (i.e., there are $n + 1$ different performance levels). Furthermore, we scale the utility function so that $u(x_0) = 0$ and $u(x_n) = 1$. Then, the problem of generating random utilities amounts to assigning $n - 1$ random values to $u(x_1)$, ..., $u(x_{n-1})$. We only assume that the utility function is monotonically increasing in x, but no further information about its shape is available.

Kadziński and Tervonen (2013) address this problem in their Algorithm 1. This algorithm fixes the utility of the worst performance to be zero: $u(x_0) = 0$. Then it generates n random values from a uniform distribution and sorts these values in ascending order. The algorithm then assigns these values following the ranked order to $u(x_1)$, ..., $u(x_n)$. To scale the utility function, all values are finally divided by $u(x_n)$. For a monotonically decreasing function, the roles of x_0 and x_n are reversed and values are sorted in descending order.

This method has the advantage of being simple and fast to compute, but the shape of the generated utility functions may be affected by the distribution of performance levels. To show this, let us consider two attributes, ”System” and ”Flagship”, used in Kadziński and Tervonen (2013) to evaluate university systems of different countries, plus two additional attributes. Table 1 presents the performance level contained in the example (ranked from worst to best). In this table, attribute ”System” is denoted by 1, ”Flagship” is denoted by 2, and the additional new attributes are 3 and 4.

Figure 1 depicts 10 random instances of the utility functions generated according to the method described above. It is apparent that the functions generated for g_1 and g_3 have a markedly concave shape, whereas the functions generated for g_2 and g_4 have a convex shape. This bias is caused by the distribution of the different performance values in the set of alternatives (it is noteworthy to

1	2	3	4
37	8	0	0
38	36	2	25
39	40	4	50
40	44	6	65
41	48	9	75
42	60	12	83
44	68	17	88
45	72	25	91
47	80	35	94
50	84	50	96
51	88	75	98
59	90	100	100
74	96		
82			

Table 1: Sorted values for four attributes

mention that the functions generated for g_3 and g_4 are based on exactly the same sample of 10×11 random numbers). The difference is that performance values for g_1 and g_3 are concentrated in the lower half of the scale, whereas the performance values for g_2 and g_4 are concentrated in the upper half of the scale.

Skewness of the distribution of performance values thus leads to a bias in the generated utility function. If the distribution is skewed to the right, the average distance between two performance values decreases as performance values increase. If n utility values are drawn from a uniform distribution, the expected difference between any two neighboring values is $1/(n+1)$. The smaller the difference between two performance values, the larger is the derivative of the utility function. Thus, a right skewed distribution of performance levels leads to a convex shape of the utility function.

In many applications, it is likely that the distribution of performance levels is skewed to the right. If alternatives are plans for actions developed by (groups of) decision makers, one will try to find alternatives which perform well in a majority of attributes, although one has to accept worse performance in the remaining attributes. If performance levels of all alternatives are different, the problem will contain more good than weak performance levels in each attribute, leading to a right-skewed distribution.

A way to avoid this bias is to add additional elements to the vector x, which do not correspond to the performance of any alternative in A, so that the distribution of performance levels is balanced. However, this causes the generated functions to be biased towards a linear shape. As more points are added, the generated functions tend to become very similar to each other (variance is lost). This is illustrated in Figure 2. The situation on the left corresponds to criterion g_3, adding points so that the difference of performance between consecutive elements in x is at most 5. This solves the concavity bias, but now the function tends to a linear form. If we added even more points, we could reach the situation depicted on the right part of the figure, corresponding to a step of two. There is no apparent bias, but the variance of utility values assigned to each performance level is drastically reduced.

Another straightforward approach to generate a random utility function is to randomly generate

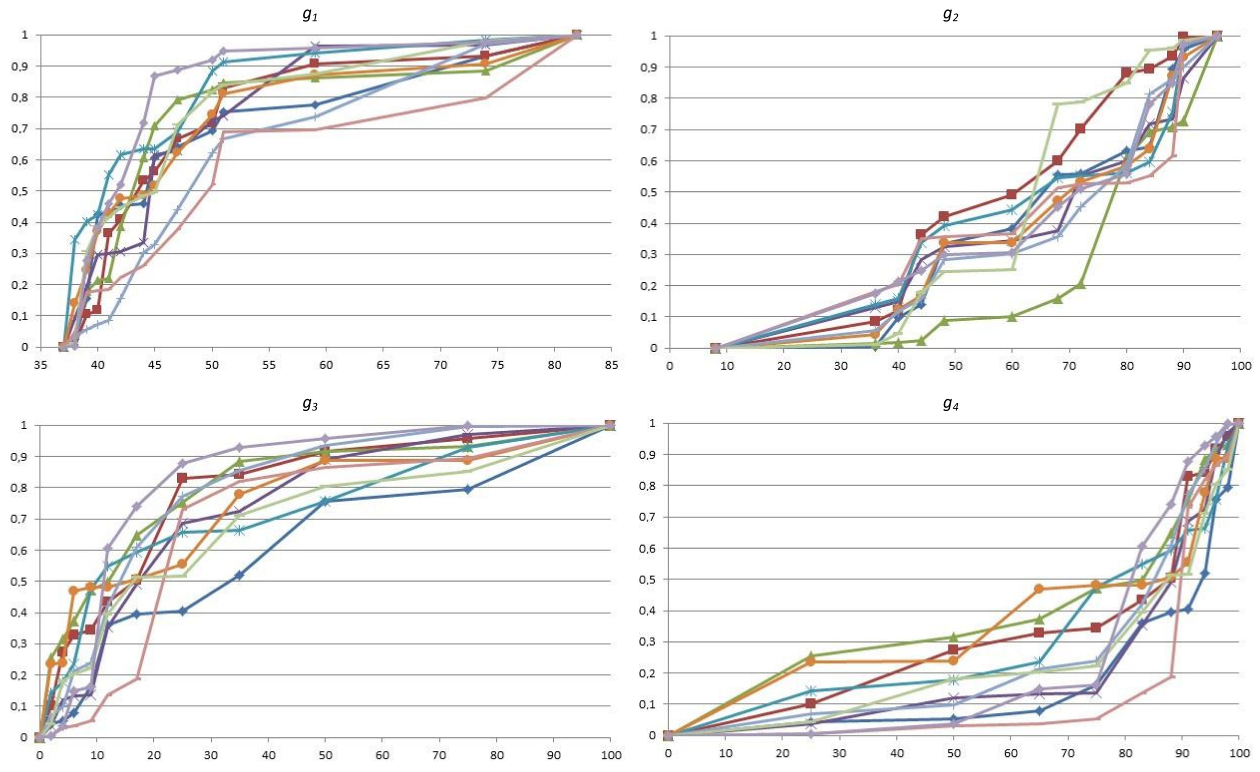


Figure 1: Utility functions generated by ranking random values

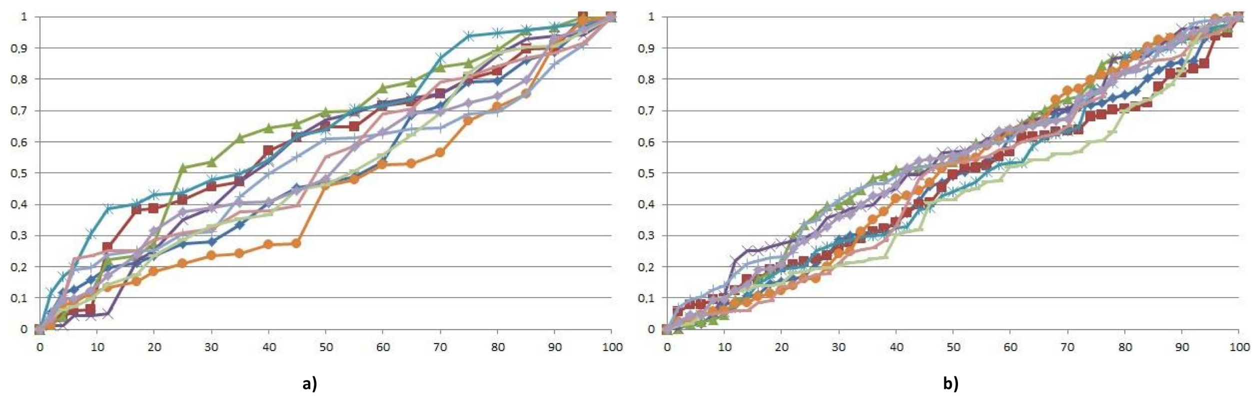


Figure 2: Utility functions generated by ranking random values with added performance points: a) points added with a step of 5; b) points added with a step of 2.

utility increases between consecutive performance levels. In such a method, one would fix $u(0) = 0$, generate n random numbers $r_1, ..., r_n$, and then successively set $u(x_i) = u(x_{i-1}) + r_1$ for $i = 1, \dots, n$. All the utilities could be divided by $u(x_n)$ at the end to scale the utility function between 0 and 1. Unfortunately, as our computational experiments will show, this approach suffers from the same issues as the approach of directly generating and sorting utilities.

4. Alternative methods

4.1. Bisection approach

In the preceding section, we have identified two main problems of previous approaches:

1. An unbalanced distribution of attribute values creates a bias towards a concave or convex utility function, depending on where the majority of performance levels (the attribute values) is located.
2. Sorting or gradually building up utility values reduces the variance as the number of points increases, leading to only a narrow band of relatively similar utility functions.

The first problem could be circumvented by introducing equally spaced points, but this worsens the second problem. Any approach which processes attribute values sequentially in ascending or descending order will necessarily limit the possible range of utility values more and more the further it proceeds. Thus, to avoid convergence to a narrow band, utility values should not be generated sequentially in the order of performance levels. We therefore propose to use a bisection approach, which first fixes the utility function at the middle of the range to be covered, then in the middle of each of the resulting intervals, and so forth until the desired granularity of resolution is reached.

For convenience, we assume throughout this section that a non-decreasing utility function defined on the interval $[0, 1]$, and scaled to the same interval, is to be generated, i.e., we set $x_0 = 0 = u(x_0)$ and $x_n = 1 = u(x_n)$

```
1: procedure UtilityBisection($lower, upper, round$)
2:     $m \leftarrow (lower + upper)/2$
3:     $v \leftarrow Uniform(0, 1)$        ▷ random value from a uniform distribution in $[0, 1]$
4:     $u(m) \leftarrow (1 - v)u(lower) + vu(upper)$
5:     if $round < r$ then        ▷ iterative process ends after $r$ rounds
6:         UtilityBisection($lower, m, round + 1$)
7:         UtilityBisection($m, upper, round + 1$)
8:     end if
9: end procedure
```

Procedure UtilityBisection creates a non-decreasing utility function in the interval $(lower, upper)$ at $2^r + 1$ equally spaced points. It is started with the values $lower = u(lower) = 0$ and $upper = u(upper) = 1$. For convenience, it is formulated as a recursive procedure here, but note that the two recursive invocations in lines 6 and 7 could be executed in parallel. In fact, if the two range parameters are considered to be vectors, then it is easy to implement the procedure in a sequential way using vector arithmetic. It then would only need r iterations, and generate 2^{k-1} utility values in iteration k. By always considering the middle of the remaining interval, the procedure ensures that the maximal possible range of utility functions is covered.

4.2. Split points

A disadvantage of directly using procedure UtilityBisection is that it creates utility values for attribute levels that do not correspond to performance levels actually occurring in the problem. To obtain utility values for actual performance levels, one can use linear interpolation between the values generated by UtilityBisection. This corresponds to common practice in the elicitation of utility functions, when decision makers are asked to specify utility values only for a small number of supporting points and linear interpolation is used in between. Nevertheless, it leads to a complex trade-off. On the one hand, if r is chosen large enough so that no interval contains more than one data point (i.e., $1/2^r \leq \min(x_{i+1} - x_i)$), that might require a considerable number of data points, many of which will not be necessary if the original attribute values are not distributed evenly. This is a waste of computational resources. On the other hand, if a smaller number of data points is used, the utility values of several consecutive performance levels might fall on a straight line.

We therefore consider several variants which replace the strict bisection by an approach that utilizes an actual performance value x_s (the "split point") rather than the arithmetic midpoint m of the interval from *lower* to *upper*. This implies that the interval boundaries will always correspond to actual performance values, which we denote by x_{lo} and x_{up}. We consider the following variants for selecting such a split point:

Approximate midpoint: The actual data point x_s which is closest to the midpoint of the interval (x_{lo}, x_{up}):

$$s = \arg\min_i |x_i - (x_{lo} + x_{up})/2| \tag{1}$$

Median point: Since we assume that attribute values are sorted, the median can easily be found. However, in case the interval (x_{lo}, x_{up}) contains an even number of points, always rounding indices upwards or downwards would introduce a bias in one direction. To avoid this, the choice in case of ties has to be randomized:

$$s = \begin{cases} \lceil (lo + up)/2 \rceil & r < 0.5 \\ \lfloor (lo + up)/2 \rfloor & r \geq 0.5 \end{cases} \tag{2}$$

where r is a random number uniformly distributed in the $(0, 1)$ interval, $\lceil . \rceil$ denotes the ceiling operator (the smallest integer greater than or equal to its input) and $\lfloor . \rfloor$ denotes the floor operator (the largest integer less than or equal to its input).

Random choice and Weighted Random choice: Finally, we consider a model which selects one of the points in the interval at random. However, this might again introduce a bias if points are not evenly distributed. If attribute values are very dense in some region of the interval, it would be likely that the splitting point is also selected from that region. To counterbalance this phenomenon, we also consider a variant in which the probability that a point x_i is selected is proportional to the length of the interval between the point's two neighbors $x_{i+1} - x_{i-1}$. Thus, in the weighted variant, the probability of selecting point x_i as the splitting point between x_{lo} and x_{up} is

$$p_i = \frac{x_{i+1} - x_{i-1}}{(x_{up} - x_{lo}) + (x_{up-1} - x_{lo+1})} \tag{3}$$

The denominator of (3) results from the fact that each difference between two neighboring interior points is contained twice in the sum of intervals, since for each point we consider the distance to the next and to the previous point. The differences from the very first and very last point to their (single) neighbors are contained only once.

For all the above cases, the algorithm can be described by the following recursive procedure, where function SplitPoint implements the different variants:

```
procedure UtilityBisection2($lo, up$)
    if a performance level exists in $]x_{lo}, x_{up}[$ then
        $s \leftarrow$ SplitPoint($lo, up$)
        $u(x_s) \leftarrow$ AssignUtility($x_s, x_{lo}, x_{up}, u(x_{lo}), u(x_{up})$)
        UtilityBisection2($lo, s$)
        UtilityBisection2($s, up$)
    end if
end procedure
```

This procedure operates on predefined performance levels, which are identified via indices. The vectors of performance levels x_i and the vector of utility values assigned to them, $u_i = u(x_i)$, can thus be treated as global variables. The procedure starts with $lo = 0$ and $up = n + 1$, and stops when utility values are assigned to all of the pre-specified performance levels.

4.3. Correction for asymmetry

Since the split point selected by the approaches presented in the previous subsection is not necessarily the exact midpoint of the interval (x_{lo}, x_{up}), the shape of the resulting utility function could be biased in a similar way as in the direct generation approach. Consider the situation shown in Figure 3, where the split point is to the left of the midpoint of the interval. A point drawn from a uniform distribution in $(u(x_{lo}), u(x_{up}))$ at x_s is more likely to lie above the straight line connecting $(x_{lo}, u(x_{lo}))$ and $(x_{up}, u(x_{up}))$ than below. Therefore, the probability that the resulting shape of the utility function is concave is higher than the probability that it is convex. To correct for this problem, we introduce a *weighting approach* in which we first randomly select (with equal probability) whether the point is below or above that line, and then draw the actual value from a uniform distribution within the corresponding interval. This weighting approach is one variant of procedure AssignUtility in UtilityBisection2, the other being the direct assignment of utility values as explicitly formulated in UtilityBisection.

The weighting approach is equivalent to drawing a point from a distribution which has a step function as a density function. Let

$$w = \frac{x_s - x_{lo}}{x_{up} - x_{lo}} \tag{4}$$

The utility value for x_s is then drawn from a distribution which has a density of

$$d_{convex} = \frac{1}{2w\,(u(x_{up}) - u(x_{lo}))} \tag{5}$$

in the interval $(u(x_{lo}), u(x_{lo}) + w\,(u(x_{up}) - u(x_{lo})))$ and the density

$$d_{concave} = \frac{1}{2(1-w)\,(u(x_{up}) - u(x_{lo}))} \tag{6}$$

in the interval $(u(x_{lo}) + w\,(u(x_{up}) - u(x_{lo}))\,, u(x_{up}))$.

Figure 3 illustrates this approach. The interval (x_{lo}, x_{up}) is split at point x_s, which is not located at the midpoint of the interval. Point x_s must be assigned a utility value $u(x_{lo}) < u(x_s) < u(x_{up})$. If a point in interval C is chosen, the utility function has a con<u>C</u>ave shape, if a point from interval V is chosen, the shape is con<u>V</u>ex. To ensure that a convex and a concave shape of the utility function in the interval (x_{lo}, x_{up}) are equally likely, the utility value must be drawn from interval C and interval V with the same probability (i.e., from a distribution which assigns a probability mass of 50% to each interval).

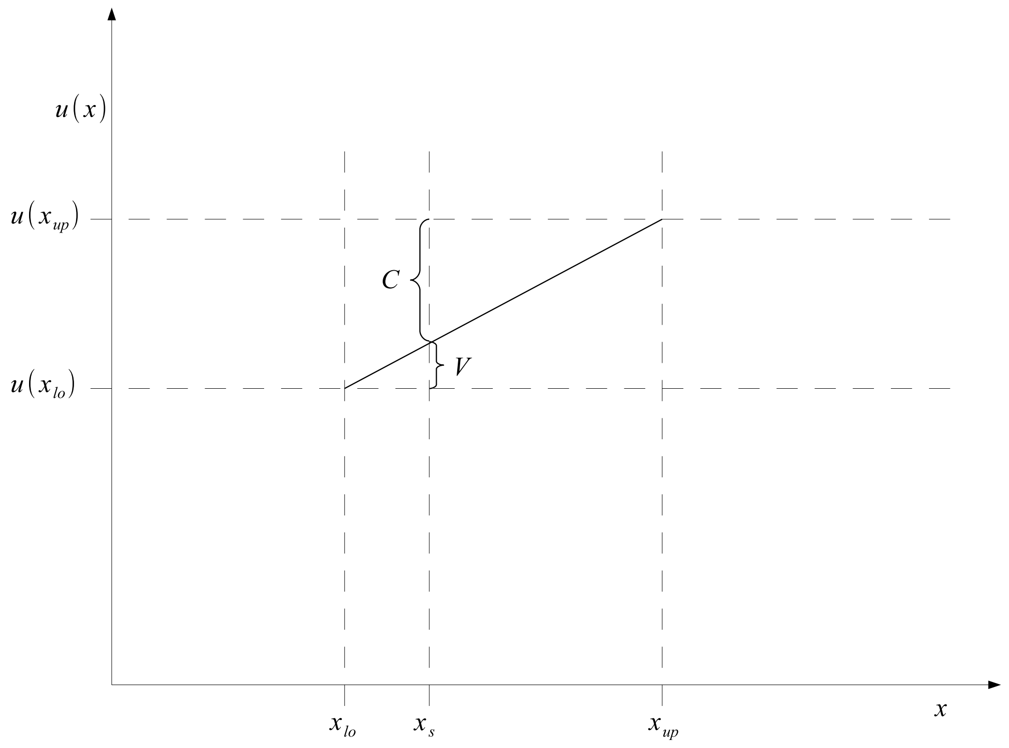


Figure 3: Adjusting the density for utility values

4.4. Didactic example

To illustrate how to use the procedures mentioned above, let us consider that for some utility function the attribute levels are $x = \{0, 24, 45, 61, 71, 80, 88, 94, 100\}$. We require the utility function to be nondecreasing, and set $u(0) = 0$ and $u(100) = 1$.

Table 2 presents the step-by-step construction of the utility function using Procedure UtilityBisection2, when the split point is determined by the approximate midpoint method. First, the closest value to the midpoint of the scale is chosen (in this case, 45 is the closest point to the midpoint of $[0, 100]$). Taking the random value r_1, one obtains $u(45) = (1 - r_1)u(0) + r_1 u(100)$. Then, the interval $[0, 45]$ is split at the point nearest its midpoint, i.e., 24. Taking the random value r_2, one obtains $u(24) = (1 - r_2)u(0) + r_2 u(45)$. Then, the interval $[45, 100]$ is split at the point nearest its midpoint, i.e., 71, and so on.

Step 1 $r_1 = 0.631$	Step 2 $r_2 = 0.741$	Step 3 $r_3 = 0.285$	Step 4 $r_4 = 0.398$	Step 5 $r_5 = 0.117$	Step 6 $r_6 = 0.241$	Step 7 $r_7 = 0.475$
$s : 2(x_2 = 45)$	$s : 1(x_1 = 24)$	$s : 4(x_4 = 71)$	$s : 3(x_3 = 61)$	$s : 6(x_6 = 88)$	$s : 5(x_5 = 80)$	$s : 7(x_7 = 94)$
$lo : 0$	$lo : 0$	$lo : 2$	$lo : 2$	$lo : 4$	$lo : 4$	$lo : 6$
$up : 8$	$up : 2$	$up : 8$	$up : 4$	$up : 8$	$up : 6$	$up : 8$
$u(x_0) = 0$	$u(x_0) = 0$	$u(x_2) = 0.631$	$u(x_2) = 0.631$	$u(x_4) = 0.736$	$u(x_4) = 0.736$	$u(x_6) = 0.767$
$u(x_8) = 1$	$u(x_2) = 0.631$	$u(x_8) = 1$	$u(x_4) = 0.736$	$u(x_8) = 1$	$u(x_6) = 0.767$	$u(x_8) = 1$
$u(x_2) \leftarrow 0.631$	$u(x_1) \leftarrow 0.468$	$u(x_4) \leftarrow 0.736$	$u(x_3) \leftarrow 0.673$	$u(x_6) \leftarrow 0.767$	$u(x_5) \leftarrow 0.744$	$u(x_7) \leftarrow 0.878$

Table 2: Didactic example of the approximate midpoint bisection method ($r_1, ..., r_7$ denote seven random numbers in $[0, 1]$, s denotes the split index, lo denotes the lower bound index, up denotes the upper bound index)

If a different SplitPoint function is chosen, the process is similar. For instance, Table 3 presents the step-by-step construction of the utility function using Procedure UtilityBisection2 when the split point is the median of the corresponding subinterval. First, the middle index, 4, is chosen. Then, the index interval $[0, 4]$ is split at index 2, and so on, using the successive random values to determine convex combinations between the utilities of the upper and lower bounds.

Step 1 $r_1 = 0.631$	Step 2 $r_2 = 0.741$	Step 3 $r_3 = 0.285$	Step 4 $r_4 = 0.398$	Step 5 $r_5 = 0.117$	Step 6 $r_6 = 0.241$	Step 7 $r_7 = 0.475$
$s : 4(x_4 = 71)$	$s : 2(x_2 = 45)$	$s : 1(x_1 = 24)$	$s : 3(x_3 = 61)$	$s : 6(x_6 = 88)$	$s : 5(x_5 = 80)$	$s : 7(x_7 = 94)$
$lo : 0$	$lo : 0$	$lo : 0$	$lo : 2$	$lo : 4$	$lo : 4$	$lo : 6$
$up : 8$	$up : 4$	$up : 2$	$up : 4$	$up : 8$	$up : 6$	$up : 8$
$u(x_0) = 0$	$u(x_0) = 0$	$u(x_0) = 0$	$u(x_2) = 0.468$	$u(x_4) = 0.631$	$u(x_4) = 0.631$	$u(x_6) = 0.674$
$u(x_8) = 1$	$u(x_4) = 0.631$	$u(x_2) = 0.468$	$u(x_4) = 0.631$	$u(x_8) = 1$	$u(x_6) = 0.674$	$u(x_8) = 1$
$u(x_4) \leftarrow 0.631$	$u(x_2) \leftarrow 0.468$	$u(x_1) \leftarrow 0.113$	$u(x_3) \leftarrow 0.533$	$u(x_6) \leftarrow 0.674$	$u(x_5) \leftarrow 0.641$	$u(x_7) \leftarrow 0.829$

Table 3: Didactic example of the median point bisection method ($r_1, ..., r_7$ denote seven random numbers in $[0, 1]$, s denotes the split index, lo denotes the lower bound index, up denotes the upper bound index)

5. Comparison

5.1. Experimental design

This section describes computational experiments (Monte-Carlo simulations) to compare the strategies described in Sections 3 and 4. The compared strategies and variants are summarized in Table 4.

Code	Description
RV	Random values, following Kadziński and Tervonen (2013) (Section 3)
RD	Random differences. Generate n random values r_i and set the utility of x_i to $u(x_i) = \sum_{j=1}^{i} r_j / \sum_{j=1}^{n} r_j$ (Section 3)
BL	Exact bisection considering 2^{k+1} points (Section 4.1) with linear interpolation in-between these points; k is the smallest integer such that $2^{k+1} > n$
DB	(Direct) approximate midpoint bisection (first method introduced in 4.2)
WB	Weighted approximate midpoint bisection, similar to DB, applying the correction presented in section 4.3
DM	(Direct) median-based bisection (second method introduced in 4.2)
WM	Weighted median-based bisection, similar to DM, applying the correction presented in section 4.3
DR	(Direct) bisection with random choice of splitting point using equal probabilities (third method introduced in 4.2)
WR	Weighted bisection with random choice of splitting point, applying the correction presented in section 4.3
DP	(Direct) bisection with random choice of splitting point using proportional probabilities according to (3)
WP	Weighted bisection with proportional probabilities, applying the correction presented in section 4.3

Table 4: Utility function generation methods

5.1.1. Simulation parameter

Table 5 provides an overview of the simulation parameters. For each vector of performance levels, SMAA will generate a large number n_f of random utility functions. To assess the effect of this number, we performed experiments generating 100 and 1000 functions per data vector. For brevity, we report only on the results obtained when generating 1000 utility functions. Results for the smaller sample size were qualitatively similar and can be obtained from the authors upon request.

Parameter	Values
Number of performance levels	$n \in \{10, 25, 100\}$
Skewness parameter (bias)	$\Delta \in \{0.01, 0.5, 1, 1.5, 2\}$
Number of functions to generate for each SMAA problem	$n_f \in \{100, 1000\}$

Table 5: Computational study parameters

For each of these parameter values, 1000 problems were randomly generated using the approach described in section 5.1.2. For each problem, all of the methods were run, and summary statistics on their performance as described in section 5.1.3 were collected. The entire simulation was programmed in R (the program code is available from the authors upon request).

5.1.2. Generation of artificial data

The simulation is based on artificial problems (i.e., performance vectors) in which the number and distribution of performance levels were varied. Without loss of generality, we generated performance levels in the $[0, 1]$ interval. To be able to study the problem identified in section 3, we draw performance levels from a skewed distribution with linearly increasing density. Thus, higher values occur more often and the standard ranking values approach (Section 3) will generate utility functions that tend to be convex. The skewness of the performance levels distribution is controlled by a parameter Δ, which represents the increase in density from $x = 0$ to $x = 1$. The higher the value of Δ, the more the data will bias the standard approach towards convexity.

The density function at point $x \in (0, 1)$ is

$$f(x) = d_0 + \Delta x \tag{7}$$

We have to calibrate the function so that the total probability mass is one. Since the area under a linear function in a given interval is simply the average of the values at the endpoints of the interval times the interval length (which is one in our case), we have to set

$$d_0 + \Delta/2 = 1 \tag{8}$$

and obtain

$$d_0 = 1 - \Delta/2 \tag{9}$$

Obviously, $0 \leq \Delta \leq 2$ must hold. By integrating and substituting (9) into (7), we obtain the cdf as

$$F(x) = d_0 x + \Delta x^2/2 = (1 - \Delta/2)x + \Delta x^2/2 \tag{10}$$

We use the inverse method to obtain random numbers that are distributed according to (10). We generate a random number r uniformly distributed between zero and one, and then use $x = F^{-1}(r)$ to generate the data point. Solving the quadratic equation

$$F(x) = (1 - \Delta/2)x + \Delta x^2/2 = r \tag{11}$$

for x gives, for $\Delta > 0$,

$$x = \frac{1}{2} - \frac{1}{\Delta} + \sqrt{\left(\frac{2}{\Delta} - 1\right)^2 / 4 + \frac{2r}{\Delta}} \tag{12}$$

(since $1/\Delta > 1/2$, the other solution of the quadratic equation would be negative and can be discarded). The simulation parameters consider Δ between 0.01 and 2, but if $\Delta = 0$ then equation (11) trivially yields $x = r$, corresponding to an unbiased uniform distribution in the interval $(0, 1)$.

5.1.3. Comparison criteria

The comparison of the different approaches is based on criteria that reflect two desirable properties of the utility functions to be generated for SMAA: good coverage of the sampling space and absence of biases caused by the distribution of performance levels.

- Coverage: unlike Figure 2, which displays very low variance, we wish that the utility functions generated in a SMAA analysis are sufficiently different from each other, covering the whole sampling space from extremely concave to extremely convex functions. To assess this property, we measure the fraction of the overall area of the sampling space covered by the set of utility functions generated. Note that the total sampling space has an area equal to 1 (because both the performance values and the utility values are in the interval $[0, 1]$). Let $u^*(x_i)$ (respectively, $u_*(x_i)$) denote the maximum (respectively, minimum) utility of x_i considering the n_f utility functions generated, then:

$$AreaAboveMax = \sum_{i=1}^{n} \frac{(1 - u^*(x_{i-1})) + (1 - u^*(x_i))}{2}(x_i - x_{i-1}) \tag{13}$$

$$AreaBelowMin = \sum_{i=1}^{n} \frac{u_*(x_{i-1}) + u_*(x_i)}{2}(x_i - x_{i-1}) \tag{14}$$

$$Coverage = 1 - (AreaAboveMax + AreaBelowMin) \tag{15}$$

  The more the coverage approaches 1, the better is the approach.

- Absence of biases: unlike Figure 1, we wish that the distribution of data does not cause utility functions to be biased towards convexity or concavity. We assess this according to two measures. One measure is the absolute difference between the proportion of utility function points $(x_i, u(x_i))$ that lie above the linear utility function, which is the line connecting $(0, 0)$ to $(1, 1)$, and those that lie below:

$$\begin{aligned} ShareDiff = &|\#\{(i,j) : u_j(x_i) > x_i \ (i = 2, ..., n-1; j = 1, ..., n_f)\} \\ &- \#\{(i,j) : u_j(x_i) < x_i \ (i = 2, ..., n-1; j = 1, ..., n_f)\}| \\ &/((n-2)n_f) \end{aligned}$$

  (with # denoting the cardinality of the sets). The more this difference approaches zero, the better.

  A second measure is the area under each utility function $u_j(.)$ generated for SMAA ($j = 1, ..., n_f$):

$$AreaBelow_j = \sum_{i=1}^{n} \frac{u_j(x_{i-1}) + u_j(x_i)}{2}(x_i - x_{i-1}) \tag{16}$$

We expect that a well balanced approach yields utility functions such that the average area below the function is close to 0.5 (the area below the linear utility function).

5.2. Computational results

5.2.1. Coverage

Our first criterion refers to the fact that a good method should generate widely dispersed utility functions, that cover most of the area between the utility levels of zero and one assigned to the best and worst outcome, respectively.

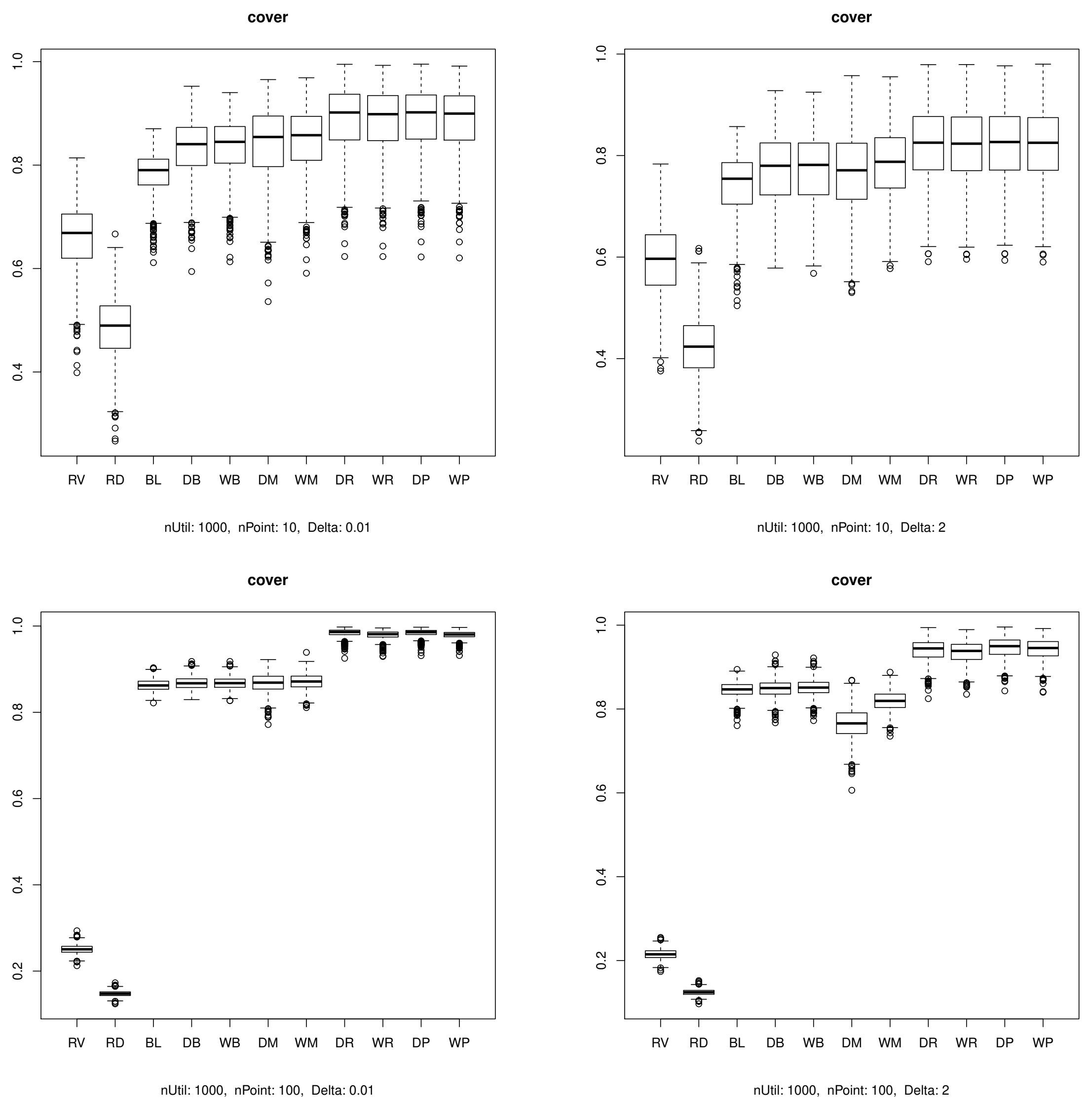


Figure 4: Distribution of coverage values for extreme simulation parameters

Figure 4 shows the distribution of the fraction of the total unit square covered by all 1000 utility functions generated for the extreme values of the simulation parameters. Increasing the skewness of the distribution of performance levels (left vs. right column in Figure 4) mainly increases the variance of this measure. Increasing the number of performance levels from 10 (top row) to 100 (bottom row) leads to a strong decrease in the variance, while the coverage on average increases for most methods.

A method is better if it provides a large coverage. According to this criterion, methods based on a random choice of a splitting point for bisection (DP, DR, WP, WR) perform best, independently of whether the point is selected using equal or proportional probabilities, and whether the correction of section 4.3 is applied or not. The next group of methods are those methods which use a splitting point close to the interval midpoint (DB, WB), or use the median value (DM, WM). Again, a correction in generating utility values does not make a large difference. In the extreme case of a very skewed distribution and generating many points on the utility function, the median-based methods fall clearly behind the midpoint based methods. Methods that generate utility values or their differences directly (RV, RD) perform consistently worse than all the other methods, and this difference still increases in larger problems.

Given the large sample size of 1000 experiments, most differences between methods are statistically significant. However, the actual differences between methods based on random selection are well below one percent, so the practical importance of these differences is low.

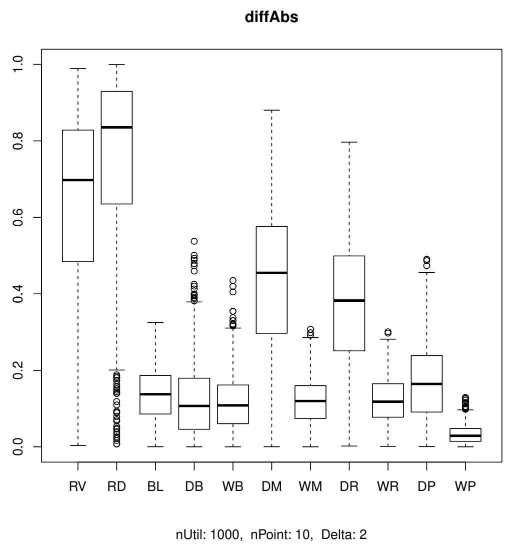
5.2.2. Shape bias

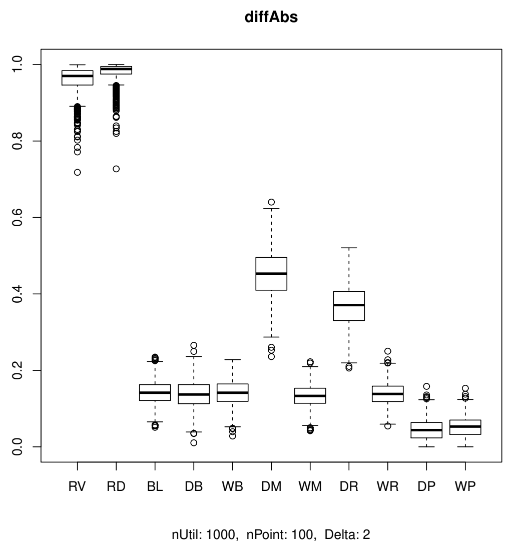
The first measure of potential bias is the difference in the share of points above and below the line that corresponds to a linear utility function (the closer to zero, the better). Figure 5 shows the distribution of this measure for highly skewed distributions of performance levels ($\Delta = 2$). For some methods, a skewed distribution of data points has a drastic effect on the shape of the utility function. Almost all utility functions generated by the direct methods RV and RD for 100 performance levels are of convex shape. In contrast, methods DP and WP perform quite well even in this extreme case. Here the weighting approach pays off, WR and WM are clearly better than DR and DM.

Figure 6 shows the effect of increasing skewness for the different methods. In particular in the case of 100 performance levels, the curves for the direct methods RV and RD approach the level of 100%, indicating that almost all generated utility functions are convex. If only 10 points are generated, this level is not yet reached, therefore the curve still increases strongly even for $\Delta = 2$. In contrast, WP and similar methods are quite unaffected by the increase in skewness.

The share difference only counts points and does not consider how far they are away from the straight line, while the average area under the utility function takes this distance into account. This puts it into direct conflict with the coverage measure, since a good coverage requires points to be far away from the central line. However, by averaging, a method which generates some functions far below and an equal number of functions far above the straight line will also perform well in that measure.

Figure 7 presents the distribution of the average area below each utility function for the same parameter settings as Figure 4. For data drawn from an almost uniform distribution (left column of Figure 7) this measure is distributed symmetrically around 0.5 (the value for a linear utility function) for all methods. Once the distribution of performance levels becomes skewed, for many methods the area decreases considerably, indicating a more convex shape. The only method that is not affected at all by this phenomenon is BL, which generates a piecewise linear utility function in a way that is completely independent of the actual performance levels. A similar robustness is achieved by WB, and at least for problems involving 100 performance levels also by WP.

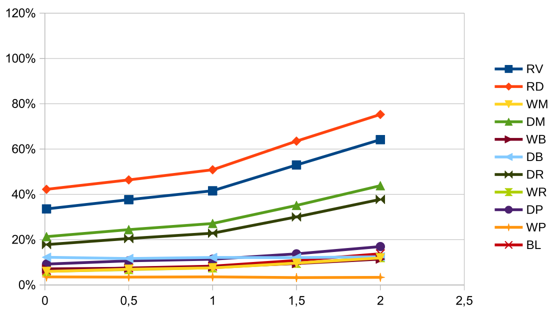


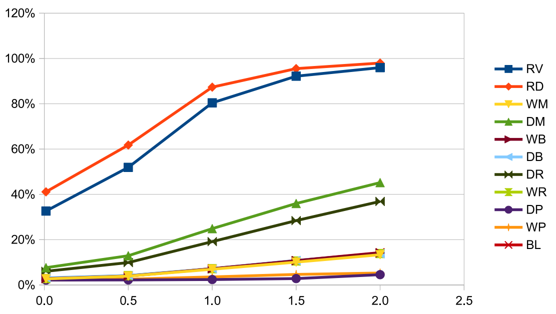


10 performance levels

100 performance levels

Figure 5: Distribution of absolute difference between share of points above and below linear utility line for highly skewed performance levels ($\Delta = 2$)





10 performance levels

100 performance levels

Figure 6: Average absolute share difference for increasing skewness

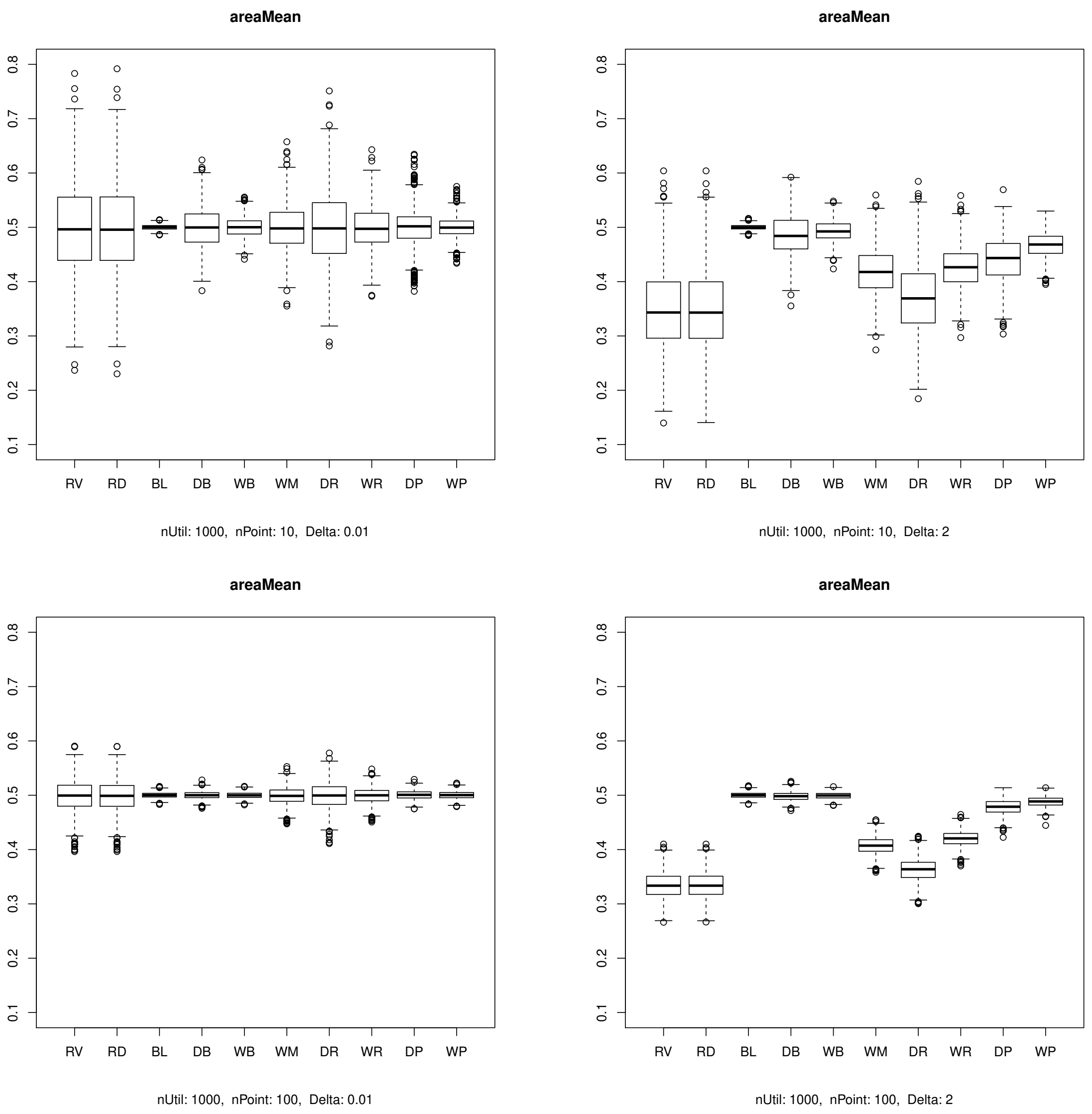


Figure 7: Distribution of area below utility function

A good method should perform well both in terms of coverage, and in terms of avoiding a bias. Since the area under the utility functions provided a very clear picture of the bias problem, we first use this measure in combination with coverage to perform a multi-dimensional evaluation of the methods.

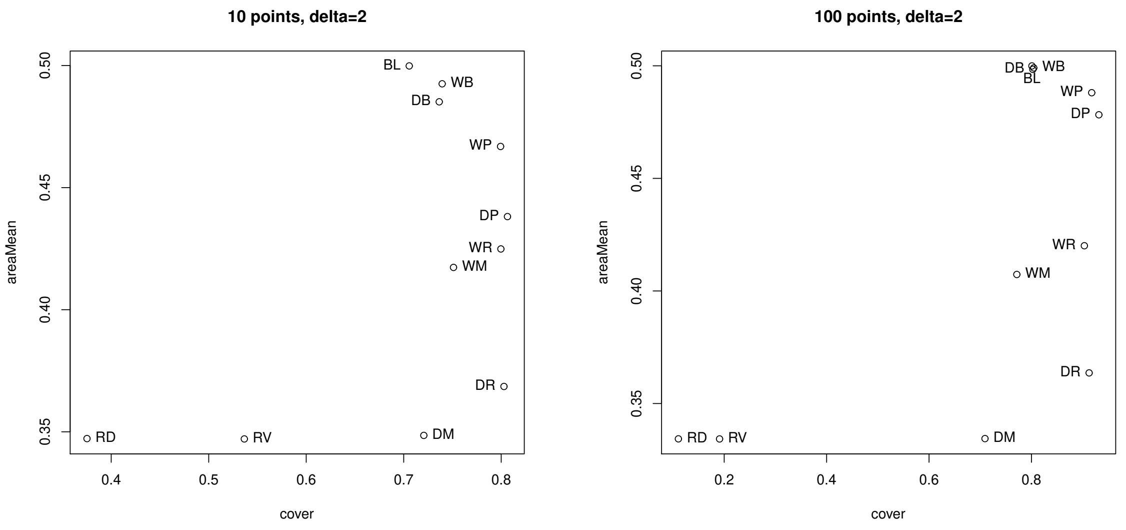


Figure 8: Average values of coverage and area under curve for different methods and highly skewed performance levels

Figure 8 visualizes possible dominance relations between methods by locating the average performance of methods in the two dimensional space formed by the two criteria. Since coverage is to be maximized, better methods are found to the right. None of the methods has an average area under the utility functions above 0.5, so for the second attribute, higher values are also better (closer to 0.5). Thus, methods in the upper right corner dominate methods below or to the left. For the case of 100 data points, BL, WB and DB have almost the same performance.

The figure clearly shows the effect of different characteristics of the methods. Correcting for the location of the splitting point consistently reduces the bias: WM is always located above DM, WR above DR, WP above DP and WB above DB. For WM vs. DM, reducing bias also leads to a higher coverage, for WP and DP, these two attributes are in conflict, for the other two pairs of methods, the effect on coverage is negligible. BL, WB, WP, and DP form the efficient frontier, with DB being very close.

A similar result is obtained by considering all three criteria, including also the share difference. Table 6 shows the sets of methods which are efficient with respect to all three criteria for all parameter combinations used in the simulation. Since values are sometimes very close, we used a threshold of 0.001 to determine whether a difference in performance values is actually relevant.

DP is the only method which is efficient in all parameter settings, mainly because it very consistently provides the best value in terms of coverage. As could already be seen in Figure 8, WP usually performs considerably better in terms of avoiding a bias. Consequently, WP is also always efficient with two exceptions of a high number of almost uniformly distributed performance levels, where the ability to avoid a bias is not so important. We therefore consider WP (and WB, which is efficient in the same settings) to be good candidates. As can already be seen from Figure

	Performance levels		
Δ	10	25	100
0.01	DP,WP	DP,WP	DP
0.5	DP,WB,WP	BL,DP,WB,WP	DP
1.0	BL,DP,WB,WP	BL,DP,WB,WP	DB,DP,WB,WP
1.5	BL,DP,WB,WP	BL,DB,DP,WB,WP	DP,WB,WP
2.0	BL,DP,WB,WP	BL,DB,DP,WB,WP	DB,DP,WB,WP

Table 6: Methods which are efficient with respect to the three criteria coverage, share difference and area for different simulation parameters

8, there is a clear trade-off between WB and WP, the former being better in terms of avoiding bias, the latter in terms of coverage.

5.3. Illustrative example

To illustrate the effects that the utilization of the proposed methods may have on the outcomes of actual decision problems, we return to the case presented in Kadziński and Tervonen (2013). For reasons of brevity, we do not reproduce the input data here. The problem is to generate a ranking of national university systems of 20 countries based on four attributes. Since we want to focus on the effect of different marginal utility functions, we furthermore assume that all attributes receive identical weights of 1/4 and we do not consider additional preference information. We compare WP, the method which performed best in avoiding bias, to the direct random value approach (RV). For this comparison, we followed the usual recommendation of SMAA methods (Tervonen and Lahdelma, 2007) and generated 10,000 utility functions for each of the attributes with both methods. We then calculated the rank acceptability indices, i.e. the probabilities that an alternative occupies a certain rank. The results of this comparison are shown in Table 7. Rank acceptability indices shown are rounded to full percentage points. An empty cell indicates an index which is exactly zero, an entry of 0 indicates a strictly positive value less than 0.5%. The first line for each country represents the values obtained when sampling utility functions with the direct method (RV), the second line the indices obtained when sampling utility functions with the WP method (proportional choice of splitting point and correction for splitting points different from the interval mean).

Two facts are noteworthy about these results. First, the situation of the first two alternatives (Germany and UK) is reversed for the two methods. Using the direct method, UK occupies rank one in about 76% of all utility functions sampled, and Germany rank one in only about 24%. For the bisection method, the corresponding probabilities are 24% and 76%, respectively. Consequently, the direct method would indicate a pairwise winning index for Germany over the UK of 24%, and the bisection method of 76%. The reason for this reversal is that many countries have low scores for the first and fourth attribute. The direct method therefore creates, on average, strongly concave partial utility functions for these attributes (e.g., function g_1 in Figure 1). In both attributes, the UK has scores which are among the best, but lower than Germany. Using strongly concave utility functions, these scores still generate a comparatively high utility (very close to Germany's). In contrast, if the bias towards concavity is corrected, functions that are closer to linear or convex shapes in that region appear more frequently, and in such cases Germany obtains on these attributes a larger utility advantage over the UK, leading Germany to surpass the UK more often.

	Rank																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
GER	24	75	0	0																
	76	23	0	0	0															
UK	76	24																		
	24	76																		
SPA			54	20	22	4	0													
			68	13	11	4	1	1	1	0	0									
SWE		0	24	29	33	14														
		0	21	30	27	21														
NET		0	21	44	30	5														
		0	7	35	45	13														
ITA			1	7	15	77	0	0	0											
		0	3	21	15	41	6	6	5	2										
FIN						0	24	55	20	2	0									
					2	11	30	31	20	5	1	0								
BEL						0	71	24	5	1	0									
					0	5	39	29	22	5	0									
AUS								3	33	35	22	7	1							
						1	4	15	28	31	14	4	3							
DEN							0	2	16	29	32	20	0							
						3	7	8	12	23	18	25	5	0						
FRA							5	17	26	25	22	6	0							
						0	14	10	7	20	26	16	5	2						
CZE								0	2	9	24	65								
								1	4	12	41	42								
POR										0	0	1	98	1						
									0	1	1	10	64	24						
SLO													1	33	44	22	0	0		
												3	21	22	31	15	5	3	0	
IRE														56	36	8	0	0		
													2	32	46	15	4	1		
HUN														11	19	67	3			
														20	18	52	10			
EST																3	77	13	8	0
																10	58	16	14	
GRE																0	11	64	24	2
																2	9	59	25	6
POL															0	0	9	23	68	
															6	5	13	19	58	
SVK																	0	0	1	98
																1	1	2	3	93

Table 7: Rank acceptability indices (in %) for ranking the university systems of 20 countries based on data from Kadziński and Tervonen (2013). Upper line: direct method (RV); lower line: weighted proportional bisection method (WP).

The second observation to be made from Table 7 is that the bisection method generates a wider range of possible ranks than the direct method. For 12 out of the 20 countries, the matrix contains a strictly positive probability for at least one more rank than in the result obtained with the direct method. One might wonder whether the fact that a method generates “less precise” results is really an advantage, but we argue that this is indeed the case. The bisection method thus alerts the decision maker to the fact that there are utility functions that could lead to these additional ranks (although often with low probabilities), the direct method with its limited sample of utility functions would (erroneously) indicate that these ranks are not possible.

6. General concave (or convex) function

So far we have considered the generation of random non-decreasing utility functions. The adaptation to non-increasing utility functions is obvious. In this section we show that the approach can also be adapted to generate utility functions of a specific shape. We illustrate our approach by considering a concave function, which might be required to represent risk aversion. The adaptation of this approach to convex, or even s-shaped, utility functions is straightforward.

Referring to Figure 3, it is obvious that to generate a function that is concave within the interval $[x_{lo}, x_{up}]$, the utility value for point x_s has to be drawn from the interval C. This will ensure that the slope of the function in $[x_{lo}, x_s]$ is larger than the slope in $[x_s, x_{up}]$. However, in order to obtain a utility function that has an overall concave shape, we must also ensure that the slope above x_{up} is smaller than the slope in $[x_s, x_{up}]$. Generating a concave utility function thus requires coordination not just within, but also across the intervals in which a bisection is made.

We therefore have to process all intervals simultaneously. We consider a situation in which the domain of the utility function is already split into m intervals j and denote the indices of the boundaries of interval j by $lo(j)$ and $up(j)$. Obviously, $up(j) = lo(j+1)$, and since the boundaries of the domain are also boundaries of intervals, $lo(1) = 0$ and $up(m) = n$.

For each interior boundary point $up(j), 0 < j < m$, we randomly generate a slope v_j between the slopes of the utility function in neighboring intervals as

$$v_j = r\frac{u(x_{up(j+1)}) - u(x_{up(j)})}{x_{up(j+1)} - x_{up(j)}} + (1-r)\frac{u(x_{up(j)}) - u(x_{lo(j)})}{x_{up(j)} - x_{lo(j)}} \tag{17}$$

where r is a random number uniformly distributed in $[0,1]$. Furthermore, we set $v_0 = \infty$ and $v_m = 0$.

Slope v_j is at the same time a lower bound on the slope of the utility function in interval j, and an upper bound on the slope of the utility function in interval $j+1$. Let $s(j)$ be the index of the split point within interval j, selected via one of the methods outlined in section 4. The utility value that can be assigned to $x_{s(j)}$ is constrained from below by the straight line between the utility values of the two boundary points:

$$u(x_{s(j)}) \geq lb_j = u(x_{lo(j)})\frac{x_{up(j)} - x_{s(j)}}{x_{up(j)} - x_{lo(j)}} + u(x_{up(j)})\frac{x_{s(j)} - x_{lo(j)}}{x_{up(j)} - x_{lo(j)}} \tag{18}$$

From above, it is bounded by the two slopes at the upper and lower boundaries of the interval:

$$u(x_{s(j)}) \leq ub_j = \min(ub_j^{lo}, ub_j^{up}) \tag{19}$$

where

$$ub_j^{lo} = u(x_{lo(j)}) + v_{j-1}(x_{s(j)} - x_{lo(j)}) \tag{20}$$

and

$$ub_j^{up} = u(x_{up(j)}) - v_j(x_{up(j)} - x_{s(j)}). \tag{21}$$

The actual utility value can then be generated randomly as

$$u(x_{s(j)}) = rlb_j + (1 - r)ub_j \tag{22}$$

where r is again a random number drawn from the $[0, 1]$ interval.

These calculations replace the assignment of a utility value in procedure UTILTYBISECTION2. Figure 9 illustrates this approach graphically.

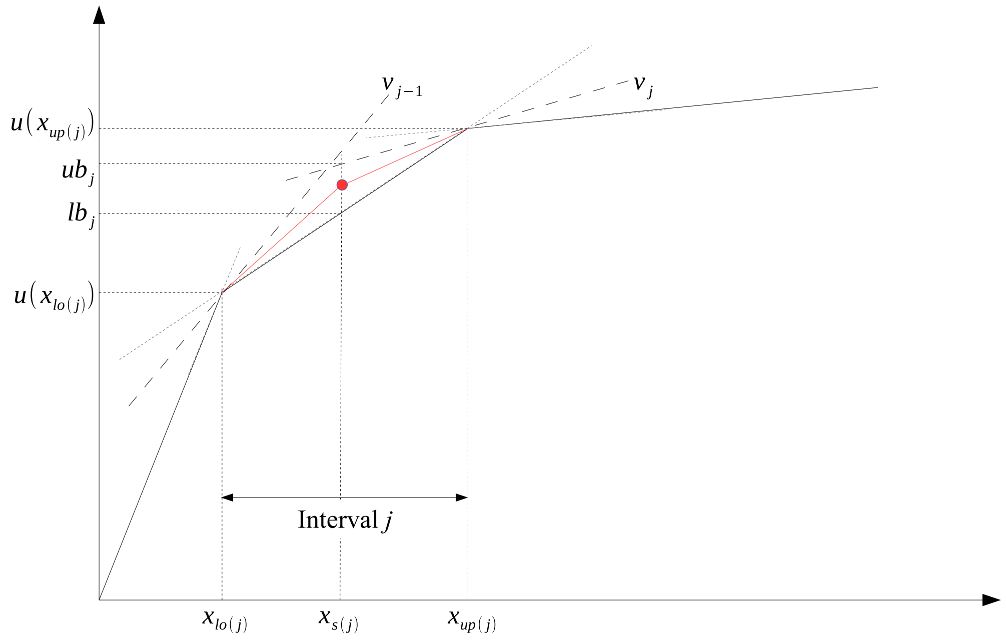


Figure 9: generating a concave utility function

A convex function can be obtained by a straightforward adaptation of this method so that the upper bound is derived from the straight line connecting boundary points, and the lower bound from the maximum of the slopes assigned to the boundary points of an interval. To generate an s-shaped function (convex for $x < x_b$ and concave for $x > x_b$) one can randomly draw an inflection point x_b (to be added to the existing ones), generate a random value for $u(x_b)$ and then use the previous processes to generate a convex value function for points x_1,...,x_b and a concave value function for points x_b,...,x_n.

All strategies to select a splitting point explained in section 4 can be applied to this method. To test the effect of the different strategies, we performed a similar simulation in which we compared the approximate midpoint (CB), the median (CM), random choice (CR) and random choice with proportional probabilities (CP) approach to selecting a split point. We also used the simulation to compare the bisection method to a naive approach (labeled CD), in which we directly generated sorted random slopes for all intervals between performance levels, and used these slopes and the differences between performance levels to successively calculate the utility values.

		$\Delta = 0.01$			$\Delta = 2$		
Points	Method	above	below	coverage	above	below	coverage
10	CD	20.60	53.50	25.91	27.24	53.05	19.71
	CB	11.67	50.03	38.30	17.22	50.03	32.75
	CM	10.71	50.03	39.26	21.11	50.02	28.88
	CR	5.57	50.02	44.41	14.93	50.01	35.06
	CP	5.58	50.02	44.40	14.85	50.02	35.13
100	CD	29.16	62.32	8.52	34.24	59.41	6.35
	CB	11.28	50.03	38.69	11.88	50.03	38.09
	CM	11.34	50.03	38.63	22.60	50.02	27.39
	CR	0.82	50.02	49.16	5.49	50.01	44.50
	CP	0.83	50.02	49.16	4.78	50.02	45.20

Table 8: Area above and below and utility function and coverage for methods to generate a concave function (in %)

Since our aim is to generate a concave function, the measures used to indicate bias for a certain shape are not meaningful here. We therefore focus on coverage, but also analyze the fraction of the unit square that is below, and above each set of 1000 generated utility functions. Obviously, since we generate concave functions, the area below any utility function must be at least 50% of the unit square.

Results of this simulation are shown in Table 8. All methods based on bisection provide a considerably larger coverage than the direct approach, and as in the case of the general monotonic functions, selecting the split point at random with proportional probabilities (CP) is the best approach. Interestingly, the area below any utility function is very close to 50% for all bisection methods. This indicates that these methods always generated at least one function (out of 1000) that was almost linear. Only the naive approach leaves a larger gap between all generated utility functions and the straight line. The advantage of random strategies (CR and CP) results mainly from the fact that they also generate some utility functions close to the upper border of the unit square, and thus leave only a small area above the highest utility values not covered.

7. Conclusions

As we have shown in this paper, the generation of utility values for given performance levels in SMAA is not a trivial problem. If the distribution of performance levels is skewed, direct assignment of utility values to performance levels can lead to a bias towards a specific shape (convex or concave) of the resulting utility function.

We therefore propose an approach to correct for this problem and to ensure that generated utility functions are balanced between different shapes, and cover a large part of the possible universe of such functions. Our approach is based on a bisection method, and therefore requires a method to split the domain of the utility function into intervals. We tested several methods for this split in a computational study, and found that methods that select a split point randomly and adjust selection probabilities according to the distance between performance levels perform best. We also developed methods to correct for the fact that the split point is not in the middle of the interval. Furthermore, we have shown how this method can be modified to generate utility functions of a specific shape.

Our computational results indicate that the methods we propose perform considerably better than direct approaches, both in terms of covering the entire set of possible functions, and in terms of generating a balanced set of utility functions of different shapes. Of course, more elaborate algorithms also require more computational resources. For generating utility functions of arbitrary shape, the additional effort of our methods is not very high. Only few additional calculations are required to find the boundaries between which utility values must be placed. Furthermore, the process can be parallelized to a large extent, which is particularly important when implementing it in vectorized languages like R. Actual implementation on parallel processors is also possible, but probably not necessary given the low computational effort.

Our research presented in this paper can be extended in several ways. We have validated our approach with a computational study in a specific setting. In particular, we used one specific distribution (a linearly increasing density function) to generate the skewed distribution of performance levels needed to study the bias towards a convex utility function. It is not yet clear how robust our results in terms of the ranking of methods are with respect to this assumption. Further studies could utilize different distributions of performance levels to study whether a bias towards a convex utility function can be robustly avoided also if performance values are distributed differently. This line of research could be complemented with empirical studies indicating how performance levels are distributed in real decision problems. Once these distributions are known, it would be possible to study which methods are most appropriate to avoid biases in such a real life setting. Still, as our results indicate, the methods we have proposed are at least an important step in creating more balanced and thus more informative input data for SMAA or other approaches that may use utility functions (e.g., decision trees, negotiation analysis).

Acknowledgements

L. Dias acknowledges the support of FEDER and Fundação para a Ciência e Tecnologia (FCT) through projects POCI-01-0145FEDER-016765 and MIT-EXPL/SUS/0013/2017

References

Cinelli, M., Coles, S., Nadagouda, M., Błaszczyński, J., Słowiński, R., Varma, R., and Kirwan, K. (2017). Robustness analysis of a green chemistry-based model for the classification of silver nanoparticles synthesis processes. *Journal of Cleaner Production*, 162:938–948.

Corrente, S., Figueira, J., Greco, S., and Słowiński, R. (2017). A robust ranking method extending ELECTRE III to hierarchy of interacting criteria, imprecise weights and stochastic analysis. *Omega*, 73:1–17.

Greco, S., Ishizaka, A., Matarazzo, B., and Torrisi, G. (2018). Stochastic multi-attribute acceptability analysis (SMAA): an application to the ranking of Italian regions. *Regional Studies*, 52(4):585–600.

Hokkanen, J., Lahdelma, R., Miettinen, K., and Salminen, P. (1998). Determining the implementation order of a general plan by using a multicriteria method. *Journal of Multi-Criteria Decision Analysis*, 7(5):273–284.

Jacquet-Lagrèze, E. and Siskos, J. (1982). Assessing a set of additive utility functions for multicriteria decision-making, the UTA method. *European Journal of Operational Research*, 10:151–164.

Kadziński, M., Greco, S., and Słowiński, R. (2012). Selection of a representative value function in robust multiple criteria ranking and choice. *European Journal of Operational Research*, 217:541–553.

Kadziński, M. and Tervonen, T. (2013). Robust multi-criteria ranking with additive value models and holistic pair-wise preference statements. *European Journal of Operational Research*, 228(1):169 – 180.

Kadziński, M. and Michalski, M. (2016). Scoring procedures for multiple criteria decision aiding with robust and stochastic ordinal regression. *Computers and Operations Research*, 71:54–70.

Kirppu, H., Lahdelma, R., and Salminen, P. (2018). Multicriteria evaluation of carbon-neutral heat-only production technologies for district heating. *Applied Thermal Engineering*, 130:466–476.

Lahdelma, R., Hokkanen, J., and Salminen, P. (1998). SMAA - stochastic multiobjective acceptability analysis. *European Journal of Operational Research*, 106(1):137 – 143.

Lahdelma, R., Miettinen, K., and Salminen, P. (2003). Ordinal criteria in stochastic multicriteria acceptability analysis (SMAA). *European Journal of Operational Research*, 147(1):117 – 127.

Lahdelma, R. and Salminen, P. (2001). SMAA-2 : Stochastic multicriteria acceptability analysis for group decision making. *Operations Research*, 49(3):444–454.

Lahdelma, R. and Salminen, P. (2010). Stochastic multicriteria acceptability analysis (SMAA). In Ehrgott, M., Figueira, J., and Greco, S., editors, *Trends in multiple criteria decision analysis*, pages 285–315. Springer.

Lahdelma, R. and Salminen, P. (2012). The shape of the utility or value function in stochastic multicriteria acceptability analysis. *OR Spectrum*, 34(4):785–802.

Li, K., Yuan, S., Wang, W., Wan, S., Ceesay, P., Heyse, J., Mt-Isa, S., and Luo, S. (2018). Periodic benefit-risk assessment using Bayesian stochastic multi-criteria acceptability analysis. *Contemporary Clinical Trials*, 67:100–108.

Loikkanen, O., Lahdelma, R., and Salminen, P. (2017). Multicriteria evaluation of sustainable energy solutions for colosseum. *Sustainable Cities and Society*, 35:289–297.

Moskowitz, H., Preckel, P., and Yang, A. (1993). Decision analysis with incomplete utility and probability information. *Operations Research*, 41(5):864–879.

Sarabando, P., Dias, L. C., and Vetschera, R. (2019). Group decision making with incomplete information: a dominance and quasi-optimality volume-based approach using Monte-Carlo simulation. *International Transactions in Operational Research*, 26(1):318–339.

Stewart, T. J. (1996). Robustness of additive value function methods in MCDM. *Journal of Multi-Criteria Decision Analysis*, 5(4):301–309.

Tervonen, T. and Figueira, J. R. (2008). A survey on stochastic multicriteria acceptability analysis methods. *Journal of Multi-Criteria Decision Analysis*, 15(1-2):1–14.

Tervonen, T. and Lahdelma, R. (2007). Implementing stochastic multicriteria acceptability analysis. *European Journal of Operational Research*, 178(2):500 – 513.

Tonin, F., Steimbach, L., Borba, H., Sanches, A., Wiens, A., Pontarolo, R., and Fernandez-Llimos, F. (2017). Efficacy and safety of amphotericin b formulations: a network meta-analysis and a multicriteria decision analysis. *Journal of Pharmacy and Pharmacology*, 69(12):1672–1683.

Vetschera, R. (2017). Deriving rankings from incomplete preference information: A comparison of different approaches. *European Journal of Operational Research*, 258:244–253.

Zhu, F., Zhong, P.-A., Wu, Y.-N., Sun, Y., Chen, J., and Jia, B. (2017). SMAA-based stochastic multi-criteria decision making for reservoir flood control operation. *Stochastic Environmental Research and Risk Assessment*, 31(6):1485–1497.