Fuzzy Sets and Systems

Random generation of k-interactive capacities

--Manuscript Draft--

Manuscript Number:	FSS-D-20-00321R1
Article Type:	Full Length Article (FLA)
Keywords:	fuzzy measure; capacity; Linear extension; random variate generation; k-order capacity
Corresponding Author:	Gleb Beliakov, PhD Deakin University Geelong, AUSTRALIA
First Author:	Gleb Beliakov, PhD
Order of Authors:	Gleb Beliakov, PhD
	Francisco Javier Cabrerizo, PhD
	Enrique Herrera-Viedma, PhD
	Jian-Zhang Wu

Abstract

The theory of capacities provides powerful formal methodology to account for criteria dependencies in multiple criteria decision problems. The discrete Choquet and Sugeno integrals aggregate criteria valuations accounting for criteria synergies and redundancies. We address an important problem of randomly generating capacities of special classes forsimulation studies and for capacity learning through evolutionary algorithms. We discuss two efficient methods suitable fork-interactive capacities. The results are supported bythe extensive numerical evidence and provide a useful tool for large scale simulations.

Declaration of Interest Statement

Dear Editors,

we submit our original contribution to this journal.
It is not submitted simultaneously elsewhere, all authors have checked the manuscript
and agreed to the submission. There are no conflicts of interest.

Manuscript

Random generation of k-interactive capacities

Gleb Beliakov[1,*], Francisco Javier Cabrerizo[2], Enrique Herrera-Viedma[2], and Jian-Zhang Wu[3]

[1]School of Information Technology, Deakin University, Burwood 3125, Australia
[2]University of Granada, Spain
[3]School of Business, Ningbo University, Ningbo 315211, China
[*]Corresponding author: gleb@deakin.edu.au

November 13, 2020

Abstract

The theory of capacities provides powerful formal methodology to account for criteria dependencies in multiple criteria decision problems. The discrete Choquet and Sugeno integrals aggregate criteria valuations accounting for criteria synergies and redundancies. We address an important problem of randomly generating capacities of special classes for simulation studies and for capacity learning through evolutionary algorithms. We discuss two efficient methods suitable for k-interactive capacities. The results are supported by the extensive numerical evidence and provide a useful tool for large scale simulations.
keywords: Capacity; Random simulation; k-order capacity; Linear extension.

1 Introduction

The capacity [5], which is also called fuzzy measure [3, 32], or nonadditive measure [27, 33], is a normalised monotone set function which models both criteria importances and their interactions in multiple criteria decision making context. The capacity values quantify the contributions of all subsets of criteria (called coalitions in games theory) to the decision problem, while fuzzy integrals, such as the Choquet and Sugeno integrals, aggregate the criteria valuations into one overall value used to rank the alternatives. Compared to the additive (probability) measures, which provide the weights of n individual criteria, and hence aggregate them through a weighted sum, capacities make use of non-additivity, which translates into simple logical interpretation: the sum of values of two (or more) criteria can be larger or smaller than the value of their respective coalition. We refer to [2, 16–18] for more detailed discussion of capacities.

Decision models based on capacities are sophisticated but suffer from exponential (in terms of the number of criteria n) complexity, which stems from the size of power sets. Simplifications to capacities which limit criteria interactions to subsets of small cardinality include k-additive [15], k-maxitive/minitive [24,34], k-tolerant/intolerant [23], p-symmetric [26] and k-interactive capacities [4]. These simplifications keep some criteria interactions but reduce significantly the computational complexity.

Eliciting even a reduced set of capacity values is still rather complicated, as different approaches may result in quite distinct capacities, and hence will affect the resulting rankings of the alternatives. Rather than focusing on finding one, best in some sense, capacity, one can also perform simulation studies in which the capacities are randomly sampled from some region and the respective rankings of the alternatives are studied. Sensitivity analysis can also be performed in this way.

Another application of random sampling from a set of capacities is in evolutionary optimisation. For example, when using genetic algorithms in order to find a suitable capacity matching some criteria or data, the operation of mutation requires multiple random generations of capacities. Therefore it is important to have efficient algorithms to randomly generate capacities from some particular subsets following a given distribution. The uniform distribution is the most important as it is also a basis for sampling from other distributions (for example by using acceptance/rejection approach [19]).

It turns out that random generation of capacities with uniform distribution is a very challenging problem. The difficulties are related to the high dimensionality of the space containing capacities and a complicated structure of the set of capacities. It is known [7] that the set of all capacities is a polytope with an extremely large number of vertices related to the Dedekind numbers $M(n)$ (the OEIS sequence A000372: $2, 3, 6, 20, 168, 7581, 7828354, 2414682040998, 56130437228687557907788, \ldots$), and hence it is infeasible to use convex combinations of its vertices, for example. General methods of uniformly sampling polytopes [29, 30] are also infeasible in such a high dimension.

The structure of the polytopes of capacities P, called the order polytopes, is established [6, 7, 16]. It allows one to perform the decomposition of P into simplices (of equal volume) [8], then randomly pick up one simplex and finally generate a random point within that simplex (the last step is almost trivial [13, 29]). The catch here is the extremely large cardinality of the simplicial partition, given by the OEIS sequence A046873, which reads (starting from $n = 1$) $1,\ 2,\ 48,\ 1680384,\ 14807804035657359360, \ldots$. Hence these simplices cannot be explicitly enumerated for $n > 4$.

Two very recent works provide efficient algorithms for generating randomly capacities of certain types. The MinimalsPlus method in [8] uses random walks to select simplices randomly without enumerating them explicitly. This method is applicable to both general capacities and to p-symmetric capacities. Another method is applicable to an important class of 2-additive capacities [25]. It is based on simplicial partition of that set in Möbius representation.

In this paper we focus on another type of k-order capacities, the k-interactive capacities presented in [4], which can also be used to sample k-tolerant/intolerant capacities [22] as special cases. In k-interactive capacities the values of the set function are fixed in some way for all subsets of cardinality larger than or equal to k. In particular in the way that maximises capacity entropy [21]. The criteria interactions are unrestricted in subsets of lower cardinality, but are predetermined (although not zeros as in the case of k-additive capacities) for subsets of cardinality higher than k. This way the complexity of capacities is reduced in the same spirit as in all k-order capacities.

We provide two methods to randomly generate k-interactive capacities. Both methods rely on the relation between randomly generating a capacity and randomly selecting a linear extension in the corresponding partial order. The selected linear extension corresponds to one simplex in the simplicial partition of the (appropriately scaled) order polytope P [31], and hence we focus on selecting linear extensions uniformly.

A closely related problem is generating random monotone data sets for testing and benchmarking monotone classification and approximation algorithms was considered in a series of papers [9–12]. There weak order extensions (as opposed to linear extensions in [8]) were generated uniformly by using lattices of ideals of the posets together with Markov chains.

One method is based on the MinimalsPlus method from [8]. Here we use an initialisation procedure described in [8] followed by a random walk for a number of steps (also referred to as Markov chain). We apply this method to a modified Boolean lattice which corresponds to k-interactive capacities. The second method is based on the concept of topological sort [20], which is used to derive a linear extension of a partial order. The rationale is to improve on the initialisation step in [8] in order to have more uniformly distributed inputs to the subsequent random walk. Clearly, the better is the input, the less steps in the random walk are needed to achieve a uniformly distributed output. We apply suitable modifications after the topological sort which result in picking up a linear extension randomly, although not completely uniformly. The next step is to determine which linear extension to use based on the sampling history, which effectively makes the sample uniform. The formulation and then numerical studies of both mentioned methods constitute the main contributions of this paper.

This paper is organized as follows. After the introduction, we present some facts about the capacities, their special types and their representations. Section 3 is devoted to the random generation method based on MinimalsPlus. In Section 4 we discuss the second mentioned random sampling method in detail. Section 5 presents the numerical studies and compares the efficiency of both approaches. Section 6 contains conclusions.

2 Preliminaries

We consider the set of decision criteria $N = \{1, 2, \ldots, n\}$, $n \geqslant 2$, its power set $\mathcal{P}(N)$. Let $|S|$ denote the cardinality of a subset $S \subseteq N$. The definitions below can be found in the following references [2, 3, 8, 16, 18, 25, 35].

Definition 1. A capacity on N is a set function $\mu : \mathcal{P}(N) \to [0,1]$ such that (i) $\mu(\emptyset) = 0$, $\mu(N) = 1$; (ii) $\forall A, B \subseteq N$, $A \subseteq B$ implies $\mu(A) \leqslant \mu(B)$.

The value of $\mu(A)$ is a reflection of the decision maker's perception of the importance of this criteria subset A in the decision problem. The interaction phenomenon of multiple decision criteria can be represented by means of the Shapley simultaneous interaction index or other interaction indices [35].

The non-additivity of capacities is interpreted in this way: an additive capacity implies that the decision criteria are all independent; a superadditive (or supermodular) capacity indicates that all criteria are mutually complementary, whereas a subadditive (submodular) capacity indicates that the criteria are substitutive, redundant.

The decision maker's preference on the set of decision criteria can be expressed by adopting a particular type of capacity, such as k-additive, k-tolerant and intolerant, and k-maxitive and minitive capacity. For brevity, we do not introduce their specific definitions, which can be found in [3]. We only present the definition of k-interactive capacities, which is the topic of this contribution.

Definition 2. A capacity μ on N is called k-interactive if for some chosen $1 \leqslant k < n$ and $K \in [0,1]$

$$\mu(A) = K + \frac{|A| - k - 1}{n - k - 1}(1 - K), \text{for all } A, |A| > k.$$

Note that the formula in Definition 2 is obtained by applying the maximum entropy principle [4] which maximises the average contribution of the $n - k - 1$ smallest inputs. The k-interactive capacities significantly reduce both the number of parameters and the monotonicity constraints, and make it feasible to fit capacities to data for larger n.

For this type of capacity the values $\mu(B) = K$ for all subsets B with cardinality $|B| = k + 1$ are fixed at K. Thus the discrete derivatives, also called marginal contributions, $\Delta_i(A) = \frac{1-K}{n-k-1}$ for $|A| > k$. These conditions significantly reduce the number of variables and constraints. They also simplify the expression for the Choquet integral with respect to a k-interactive capacity

$$\begin{aligned} \mathcal{C}_\mu(\mathbf{x}) = \frac{1-K}{n-k-1} \sum_{i=1}^{n-k-1} x_{(i)} + K x_{(n-k)} \\ + \sum_{A \subseteq N, a \leqslant k} \mu(A) g_A(\mathbf{x}). \end{aligned} \quad (1)$$

We see that the contribution of the $n-k-1$ smallest inputs is averaged with the arithmetic mean while the interactions are explicitly accounted for the remaining inputs. The k-tolerant capacities arise as the special cases. The interactions in the subsets larger than or equal to k still do take place, but they are determined by the interactions in smaller subsets and the values of k, K.

The set of all capacities, which called the order polytope P, is determined by the $d = 2^n - 2$ non-negative values $\mu(A)$ and the $n2^{n-1}$ non-redundant monotonicity constraints. Note that the values $\mu(A)$ constitute a poset, which corresponds to the Boolean lattice B_n.

The structure of P was studied in various works, including [6–8, 16, 25, 26, 31]. It is known that this polytope has simplicial partition, and each simplex corresponds to a linear extension of $\mathcal{P}(N)$. A linear extension of a poset $\mathcal{P}$ is a linear order (i.e. chain) compatible with the order relations in $\mathcal{P}$.

The number of linear extensions of the Boolean lattice B_n is given by the OEIS sequence A046873, which makes it prohibitively expensive to construct the partition explicitly. On the other hand, if one has a reliable method of picking linear extensions randomly (and uniformly) without enumerating them all, then one can generate random capacities by generating random points in the unit simplex (which is by sorting the components of a random point in the unit cube and taking linear spacings), and assigning those components to $\mu(A)$ in the order of the selected linear extension. Therefore we now focus on randomly picking linear extensions.

3 Generation by MinimalsPlus method

A recently published approach for random capacity generation is based on random sampling from the set of linear extensions of a poset [8]. In its essence, there is a set of possible linear extensions of the Boolean lattice B_n, to which the capacities are related through an order-preserving function $\mu : B_n \to [0, 1]$, $\mu(0, 0, \ldots, 0) = 0$, $\mu(1, 1, \ldots, 1) = 1$. The number of such linear extensions is extremely large (OEIS sequence A046873), and their enumeration is a $\#P$-complete problem, and hence computationally infeasible. The MinimalsPlus is a method of randomly selecting a linear extension without the need to generate all of them, based on a heuristic followed by a Markov chain. The Markov chain is based on the classical Karzanov-Khachiyan chain, in which with probability 0.5 two consecutive elements at random position are swapped unless one precedes the other. The detailed description of the algorithms and a study of their computational complexity are presented in [8].

The Markov chain approach was also used in [10–12] for a related problem of generating uniformly weak order extensions (which can be seen as linear extensions on the equivalence classes of the partitions of the posets).

Once a linear extension is generated, the values of the capacities on that extension are generated as random points on a $2^n - 2$-simplex, by using a sorting procedure, see [13]. This way, provided the distribution of random linear extensions is uniform, we get the uniform distribution over the set of capacities. It is therefore sufficient to test the uniformity of the linear extensions method, namely check the experimental probability of each linear extension. This is feasible only for small n though, and then needs to be extrapolated to larger n. We verified the uniformity up to $n = 4$ and obtained encouraging results for $n = 5$ by using over 100 million trials.

Let us now look at k-interactive capacities. Recall that the capacity values for $|A| > k$ are fixed in some way, in particular as $\mu(A) = K + (1 - K)\frac{|A|-k-1}{n-k-1}$ for some number $K \in (0, 1]$ selected in advance. One special case is when $K = 1$, which corresponds to $(k + 1)$-tolerant capacities. If we replace the values $\mu(A), |A| = k + 1, \ldots, n$ with their respective equivalence classes, we can construct the partial order illustrated on Figure 1.

This partial order is suitable to the MinimalsPlus algorithm, although we do not require randomness for the values $\mu(A), |A| = k + 1, \ldots, n$, which are predetermined. Hence we take only the subset of $\mu(A), |A| = 0, 1, \ldots, k+1$ as the input to MinimalsPlus (excluding the chain in the upper part of Figure 1), generate the capacity values and then scale them linearly so that $\mu(A) = K$ when $|A| = k + 1$. We note that this order corresponds to an appropriately scaled down version of an order polytope.

The efficiency of the MinimalsPlus method depends on its initialisation, before executing the Markov chain process. Indeed, the more uniform distribution is generated at the initialisation steps, the fewer steps of the Markov chain are needed to converge to the stationary

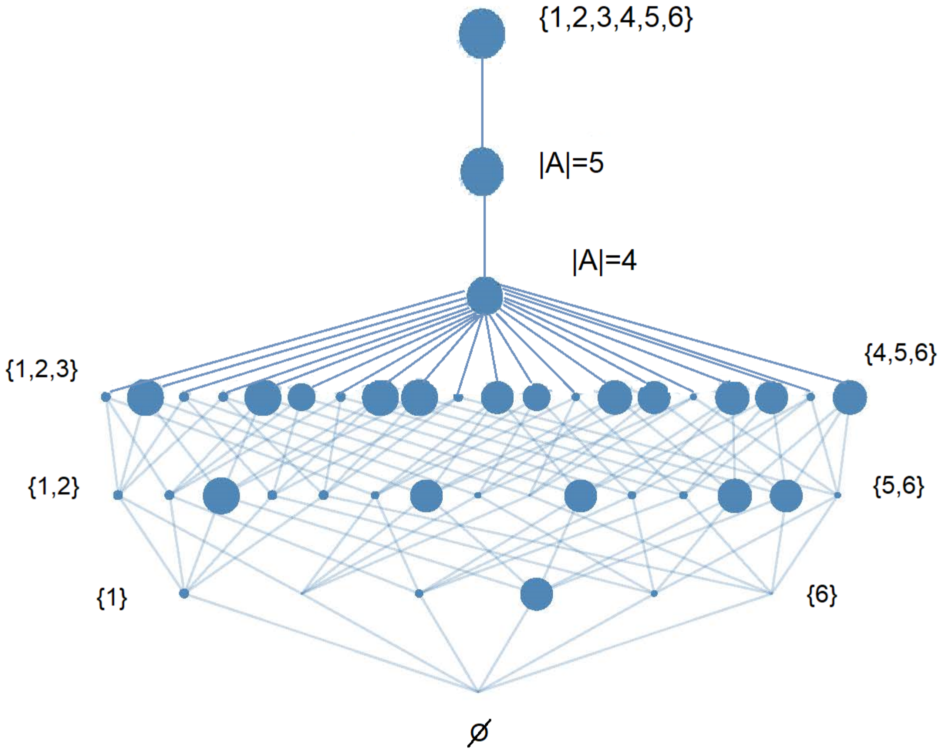


Figure 1: A 3-interactive capacity for $n = 6$ and the implied partial order. The sizes of the circles reflect the values of the respective capacities. Since all $\mu(A)$ are the same for every fixed cardinality $|A| = 4, 5, 6$, they are treated as equivalent and hence represented by single nodes.

distribution, and hence shorter CPU time. Our experiments detailed in Section 5 point to some difficulties the MinimalsPlus method encounters when generating linear extensions of some partial orders (for different combinations of n and K, hence we developed an alternative approach presented in the next section.

4 Generation by topological sort

Another way of picking a linear extension of a poset is by topological sorting [20]. Suppose we have a poset $\mathcal{P}$, or a directed acyclic graph DAG. Topological sorting is a procedure that generates a linear order such that if there is an edge between vertices u and v, then $u \leqslant v$ in that ordering. The oldest topological sort method is due to Kahn [20], which is implemented in most unix-based computer systems, as a useful routine to perform objects linking. A different algorithm is based on depth-first search from the nodes of the DAG in arbitrary order.

For our purposes Kahn's method is not suitable, as it consistently generates almost the same linear extensions, and hence we focused on the depth-first search. We construct the DAG represented by the list of pairs of vertices related by inclusion in the power set $\mathcal{P}(N)$, and excluding those pairs easily obtained by transitivity. We ensure that the algorithm is always presented the nodes in arbitrary order by randomising the input pairs. This way we hope to obtain a variety of linear extensions as the outputs of the depth-first topological sort.

While indeed the linear extensions generated in that way seemed random, by no means they were representative of the whole set of linear extensions. The topological sort algorithm used in our studies has definite preference for a particular (although large) subset of linear extensions. We detail how we performed the tests in Section 5.

In order to obtain wider variety of linear extensions we used the following result: If the DAG allows for more than one topological ordering, and one has found one ordering, then it is possible to form the second ordering by swapping two consecutively placed vertices not connected by an edge. It is therefore possible to construct up to $O((2^n - 2)!)$ other linear extensions from one linear extension found by the topological sort procedure.

As a matter of fact, the Markov chain in the MinimalsPlus method is also based on this observation and in that method two consecutive elements of the current linear extension are swapped (with some probability) unless this move contradicts the partial order. Similarly we also execute the Markov chain as in MinimalsPlus, although we do that for fewer steps.

The presented approach allows us to obtain a much broader variety of linear extensions. But it cannot be used directly for capacity generation. Firstly, the distribution is still non-uniform (although much closer to being uniform than by using topological sort on its own), and secondly, the linear extensions generated in that way are correlated.

To deal with these two issues we took the following approach. Let us make a record of the linear extensions already generated, including the number of times each extension was generated. To check how many times a given extension was generated we can compute the hash function of the linear extension and maintain the history in an array of frequencies of the generated linear extensions. Thus we can now decide whether the algorithm should return a linear extension based on its frequency, and hence we can make the distribution of outputs more uniform (in fact this is very similar to acceptance/rejection methodology). Secondly, we do not accept all linear orders obtained by swapping the consecutive vertices in the linear extension after one topological sort (as they are correlated), but only one of them, specifically the one which has not been seen before. Thus every topological sort (initiated from a random node) results in just one randomly chosen linear extension.

Let us formalise this algorithm.

Remark 1. The Algorithm 1 is simplified for readability. It assumes the array H is maintained by indexing its entries through the hash value of S, and hence the complexity of accessing $H(S)$ is $O(1)$. In C++ standard library there is data structure for this called `unordered_map`. Algorithm 1 also uses a method of returning an item with a given probability, without knowing how many items to choose from there are. It is called reservoir weighted sampling [14], which returns a sample of size k from population of size m, not knowing m in advance in a single pass. Its expected running time is $O(k \log(m))$.

Remark 2. If we are not concerned about correlated linear extensions, we can make the Algorithm 1 return a sample of linear extensions of size k after each topological sort. The linear extensions can be stored and then the population is randomised, thus breaking any

Algorithm 1: Random generation of linear extension

Input: Partial order $\mathcal{P}$

Output: Random linear extension

Assumption: The history of previously generated linear extensions H maintained as an array indexed by hash values.

Step 1. Construct the DAG by listing the pairs of elements of $\mathcal{P}$ related directly (not by transitivity). Randomise the list of pairs.

Step 2. Perform topological sort on DAG followed by Markov chain. The output is array S.

Return S with probability proportional to $\frac{1}{H(S)+1}$

Step 3. **for** k *in* $1 : |\mathcal{P}| - 1$ **do**

  if *pair* $(S[k], S[k+1])$ *not in DAG* **then**

    Swap$(S[k], S[k+1])$ `/* now` S `is another linear extension */`

    return S with probability proportional to $\frac{1}{H(S)+1}$

    Swap$(S[k], S[k+1])$ `/* swap back */`

  end

end

correlations. That would increase its efficiency by performing topological sort less frequently.

Remark 3. A different expression for probability of accepting a linear extension can be used. The square of the expression in Algorithm 1 will result in more aggressive equalising the frequencies $H(S)$. In our experience that was necessary only for $n < 5$, as for larger n the number of possible linear extensions grows extremely quickly and almost all values $H(S)$ were 1.

5 Numerical experiments

In this section we describe our numerical experiments benchmarking the two presented methods of generating linear extensions randomly and uniformly for a variety of partial orders identified by parameters n and k. Specifically we measure the quality of the resulting distribution and numerical efficiency of each method through CPU time.

To measure the quality of the distribution we calculate its statistical distance from the uniform distribution. When the number of the linear extensions L is known, that calculation is simple

$$D = \sum_{i=1}^{T} \left| \frac{1}{L} - \frac{H(S_i)}{T} \right|,$$

where T is the total number of generated linear extensions and $H(S_i)$ is the number of times the linear extension S_i was returned by the algorithm. For example for any n and $k = 2$ we always have $L = n!$, and for $k = n - 1$ L is given by the OEIS sequence A046873. However in most cases the number of linear extensions will not be known, or can be so large that it is not readily computable (compare to the OEIS sequence A046873). In the case of unknown but relatively small L we approximate it experimentally by taking the largest number found by any of our methods in a long run ($T \sim 10^8$). But when L is larger than T our expectation is to observe most, if not all, $H(S_i) = 1$, which would point to some degree of uniformity of the resulting distribution, and hence we take $L = T$.

Our experiments were performed on the following hardware: Linux workstations with 3.2 GHz Intel 64-bits processor, 32 GB of RAM. The results are summarised in Table 1. As we can see, for relatively small problems both methods presented perform similarly. Although we should mention that we needed to use longer Markov chain in MinimalsPlus method in order to get an acceptable closeness to uniform distribution. The topological sort initialisation required fewer subsequent Markov steps. Shorter Markov chains (about 100 steps) resulted in a significant degradation of the quality of the distribution generated by MinimalsPlus, but did not drastically affect that of the topologial sort.

For larger problems with the numbers of linear extensions well above 10^{10} we could not verify closeness to uniform, but we observed that each of the 10^8 linear extensions was generated once only, hence pointing toward uniformity.

We observed that applying the rejection step in Algorithm 1 (Step 3) based on history was largely redundant, and that the output of the topological sort followed by just 100 Markov chain steps was sufficient. The very large number of possible linear extensions L implies a very small chance the outputs of a reasonably random process can be repeated. This means that alternative methods measuring uniformity of the distribution in a polytope P, like the ones discussed in [28], should be used to verify the quality of the distribution.

In terms of CPU time, the topological sort initialisation requires fewer subsequent Markov steps, because it provides a broader set of linear orders than MinimalsPlus, and as a consequence exhibits faster generation times while maintaining due randomness.

Table 1: The CPU time in 1000s of seconds per 10^8 random samples and the distance from uniform $Dist$ ($\times 10^{-3}$) for the two methods, the length of the Markov chain is 5000 for Minplus and 1000 for Tsort, to guarantee stated distance from uniform in that range of parameters. The "?" indicates the cases where the distance cannot be found due to extremely large number of linear extensions, and where each sample was generated only once, indicating some closeness to uniform.

k	2				3				4			
method	Minplus		Tsort		Minplus		Tsort		Minplus		Tsort	
n	Dist	CPU	Dist	CPU	Dist	CPU	Dist	CPU	Dist	CPU	Dist	CPU
3	0.2	0.115	0.2	0.087								
4	2	0.118	0.7	0.089	7	13.24	4	2.1				
5	4	19.59	0.2	1.61	34	17.21	10	3.18	?	19.59	?	4
6	11	19.40	4	2.58	?	19.26	?	3.53	?	20.42	?	5.39
7	2.8	19.23	1	3.57	?	19.26	?	4.05	?	22.33	?	8.09
8	80	19.12	9	3.62	?	19.41	?	4.69	?	25.80	?	12.43
9	480	19.42	29	3.77	?	19.77	?	5.69	?	30.86	?	19.26
10	330	19.34	27	3.82	?	20.34	?	6.80	?	40.45	?	30.68
11	57	19.97	30	3.86	?	21.09	?	8.14	?	61.26	?	47.54
12	?	19.61	?	3.89	?	22.26	?	9.72	?	81.15	?	72.88

6 Conclusions

We proposed two methods for random generation of k-interactive capacities, based on generation of linear extensions for a specified partially ordered set. The methods differ in their initialisation step. The first method is based on the recently reported MinimalsPlus method, and the second one is based on the topological sort procedure. Both methods are followed by a Markov chain until the distribution becomes uniform. The required length of the Markov chain is smaller for the topological sort based procedure, and hence this method has better computational efficiency. The extensive numerical experiments confirmed the uniformness of the resulting distributions. Applications of the methods presented are envisaged in simulation studies and capacity learning by stochastic optimisation.

There are also other problems and types of capacities to which the presented methods can be extended. The $p-$symmetric capacities [26] and sparse capacities (where only a limited number of Móbius values are non-zero) are two immediate cases. Random generation of supermodular capacities (which form a subset of an order polytope) also used a similar procedure [1]. Finally, random generation of monotone data sets [10–12] can also benefit from the proposed technique.

Acknowledgements

The work was supported by the National Natural Science Foundation of China (No. 71671096), the K.C.Wong Magna Fund in Ningbo University and by the Spanish Ministry of Economy and Competitiveness with FEDER funds (grant TIN2016-75850-R).

References

[1] G. Beliakov. On random generation of supermodular capacities. *IEEE Transactions on Fuzzy Systems*, doi: 10.1109/TFUZZ.2020.3036699:1–6, 2021.

[2] G. Beliakov, H. Bustince Sola, and T. Calvo. *A Practical Guide to Averaging Functions*. Springer, New York, 2016.

[3] G. Beliakov, S. James, and J.-Z. Wu. *Discrete Fuzzy Measures: Computational Aspects*. Springer, Cham, Switzerland, 2020.

[4] G. Beliakov and J.-Z. Wu. Learning fuzzy measures from data: simplifications and optimisation strategies. *Information Sciences*, 494:100–113, 2019.

[5] G. Choquet. Theory of capacities. *Annales de l'institut Fourier*, 5:131–295, 1954.

[6] E. Combarro and P. Miranda. Identification of fuzzy measures from sample data with genetic algorithms. *Computers & Operations Research*, 33(10):3046–3066, 2006.

[7] E. F. Combarro, I. Díaz, and P. Miranda. On random generation of fuzzy measures. *Fuzzy Sets & Systems*, 228(4):64–77, 2013.

[8] E. F. Combarro, J. Hurtado de Saracho, and I. Díaz. Minimals plus: An improved algorithm for the random generation of linear extensions of partially ordered sets. *Information Sciences*, 501:50–67, 2019.

[9] B. De Baets, H. De Meyer, and K. De Loof. On the cycle-transitivity of the mutual rank probability relation of a poset. *Fuzzy Sets and Systems*, 161:2695–2708, 2010.

[10] K. De Loof, B. De Baets, and H. De Meyer. On the random generation and counting of weak order extensions of a poset with given class cardinalities. *Information Sciences*, 177:220–230, 2007.

[11] K. De Loof, B. De Baets, and H. De Meyer. On the random generation of monotone data sets. *Information Processing Letters*, 107:216–220, 2008.

[12] K. De Loof, H. De Meyer, and B. De Baets. Exploiting the lattice of ideals representation of a poset. *Fundamenta Informaticae*, 71:309–321, 2006.

[13] L. Devroye. *Non-uniform Random Variate Generation*. Springer Verlag, New York, 1986.

[14] P. S. Efraimidis and P. G. Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97:181–185, 2006.

[15] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92(2):167–189, 1997.

[16] M. Grabisch. *Set Functions, Games and Capacities in Decision Making*. Springer, Berlin, New York, 2016.

[17] M. Grabisch, I. Kojadinovic, and P. Meyer. A review of methods for capacity identification in Choquet integral based multi-attribute utility theory: Applications of the Kappalab R package. *European Journal of Operations Research*, 186(2):766–785, 2008.

[18] M. Grabisch and C. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *Annals of Operations Research*, 175(1):247–286, 2010.

[19] W. Hörmann, J. Leydold, and G. Derflinger. *Automatic Nonuniform Random Variate Generation*. Springer, Berlin, 2004.

[20] A. B. Kahn. Topological sorting of large networks. *Communications of the ACM*, 5:558–562, 1962.

[21] J.-L. Marichal. Entropy of discrete Choquet capacities. *European Journal of Operations Research*, 137(3):612–624, 2002.

[22] J.-L. Marichal. Tolerant or intolerant character of interacting criteria in aggregation by the Choquet integral. *European Journal of Operational Research*, 155(3):771–791, 2004.

[23] J.-L. Marichal. k-intolerant capacities and Choquet integrals. *European Journal of Operational Research*, 177(3):1453–1468, 2007.

[24] R. Mesiar and A. Kolesárová. k-maxitive aggregation functions. *Fuzzy Sets and Systems*, 346:127–137, 2018.

[25] P. Miranda and P. García-Segador. Combinatorial structure of the polytope of 2-additive measures. *IEEE Transactions on Fuzzy Systems*, 10.1109/TFUZZ.2019.2945243.

[26] P. Miranda, M. Grabisch, and P. Gil. p-symmetric fuzzy measures. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(supp01):105–123, 2002.

[27] E. Pap. *Null-additive Set Functions.* Kluwer Academic Pub, Dordrecht, 1995.

[28] A. Petrie and T. R. Willemain. An empirical study of tests for uniformity in multidimensional data. *Computational Statistics & Data Analysis*, 64:253–268, 2013.

[29] P. A. Rubin. Generating random points in a polytope. *Communications in Statistics-Simulation and Computation*, 13(3):375–396, 1984.

[30] R. L. Smith. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. *Operations Research*, 32(6):1296–1308, 1984.

[31] R. Stanley. Two poset polytopes. *Discrete Computational Geometry*, 1:9–23, 1986.

[32] M. Sugeno. *Theory of Fuzzy Integrals and Its Applications.* PhD thesis, Tokyo Institute of Technology, 1974.

[33] J.-Z. Wu and G. Beliakov. Nonadditivity index and capacity identification method in the context of multicriteria decision making. *Information Sciences*, 467:398–406, 2018.

[34] J.-Z. Wu and G. Beliakov. k-minitive capacities and k-minitive aggregation functions. *Journal of Intelligent and Fuzzy Systems*, 37(2):2797–2808, 2019.

[35] J.-Z. Wu and G. Beliakov. Nonmodularity index for capacity identifying with multiple criteria preference information. *Information Sciences*, 492:164–180, 2019.

Authors' responses to the comments on FSS-D-20-00321

November 13, 2020

Ref.: Ms. No. FSS-D-20-00321 entitled "Random generation of k-interactive capacities" (by Gleb Beliakov, et al) submitted to *Fuzzy Sets and Systems.*

The authors thank the reviewers and the AE for their very useful comments. Below we detail some of the changes made to the manuscript addressing the reviewers' concerns.

AE:

Two experts have reviewed your manuscript. Both have a very positive opinion and recommend its acceptance, provided some minor modifications are made. Reviewer 1 indicates a few misprints and typos. Rev2 mentions a formal problem, and suggests a possible way to fix it. Please, try to address their comments and submit your revised version at your earliest convenience. Thanks for your submission!

Thank you, we have thoroughly revised the manuscript. We also added additional relevant references.

Reviewer #2:

The paper is written clearly with almost no problems. It presents two methods for generating linear extensions of posets with improved heuristic to generate these extensions (more) uniformly. (The extensions are then used to generate k-interactive capacities.) The methods are tested by numerical experiments. I recommend it for publication in Fuzzy Sets and Systems journal.

(Numbering of lines provided in the manuscript is shifted. Please count it directly from text.)

1) Dedekind numbers are/is A000372 OEIS sequence. (Could be mentioned in the paper.)

2) p. 2, l. -10: the number starting 561. . . should be splitttted. (Typography.)

3) p. 2, l. -5 – -4: simple (the; (no space between (and the); (Typography.)

4) p. 4, l. 9: larger than k ; (not or equal to, see Definition 2)

5) One line before Preliminaries section: Section 6 contains conclusions.

6) p. 6, l. 12: parenthesis, $(k + 1)$-tolerant ;

Thank you, all the issues have been corrected.

Reviewer #2:

The authors introduce two procedures for the random generation of k-interactive capacities based on two different ways of randomly generating linear extensions of a poset: the so-called MinimalsPlus method and an algorithm based on topological sorting. The paper is well-written and motivated, clear and easy to read. The numerical simulations presented by the authors support their claim that the method based on topological sorting is superior to MinimalPlus on the particular posets considered in the paper.

However, there is a small modification that should be made before the paper is ready for publication. The method proposed in the manuscript for generating k-interactive capacities from linear extensions relies on the observation made by the authors that these capacities form a poset polytope. This is not strictly so. Order polytopes are known to have 0,1-valued vertices (see [1]) and this does not seem to be the case with k-interactive capacities in general. This can probably be sorted out if, for the random generation, the values of the capacity are normalised so that the first "fixed" value (for sets of size k+1) becomes 1, and then they are "scaled back". In any case, I think that this should be noted in the paper and the way of tackling the problem should be clearly explained.

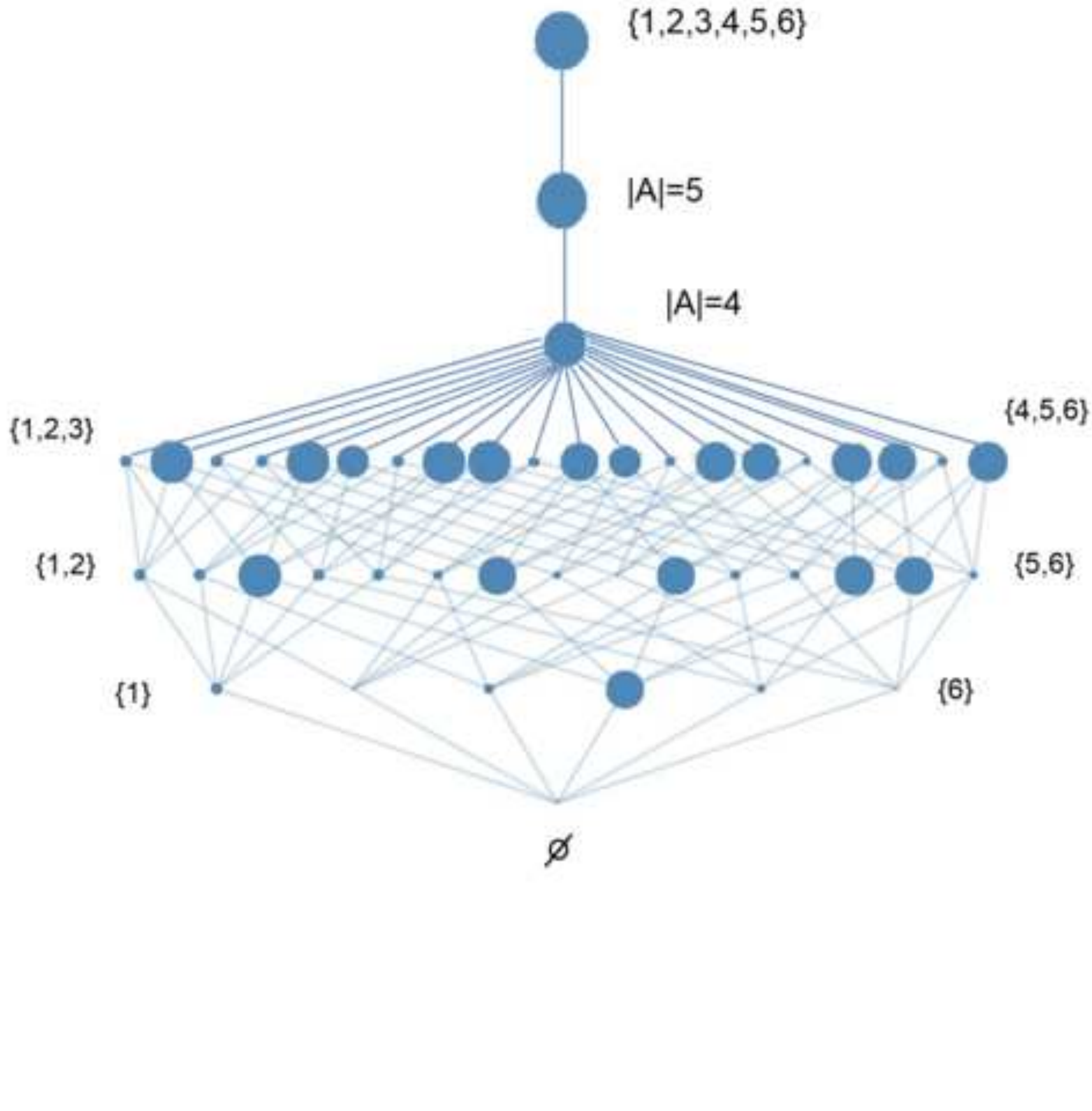
Correct, we use a scaled down version of order polytopes. We made a comment on that in the paper now.

In addition, I think it is worthwhile to mention that it would be interesting, as future work, to study if the new method of generating linear extensions works equally well for other types of posets.

Thank you, indeed the partial sort method can be used for other sorts of applications, such as generation of p-symmetric capacities and their variations, sparse capacities, and also geneation of monotonic data sets. We added several new references in respect.

[1] Two poset polytopes. Richard P. Stanley. Discrete Computational Geometry volume 1, pages 9-23 (1986)

Once again let us thank all the reviewers for their detailed comments and valuable suggestions.


{1,2,3,4,5,6}
|A|=5
|A|=4
{1,2,3}
{4,5,6}
{1,2}
{5,6}
{1}
{6}
∅