Tensor Approximation of Cooperative Games and Their Semivalues

Rafael Ballester-Ripoll

School of Human Sciences and Technology, IE University

Abstract

We propose an algorithm to approximate the semivalues of general transferable-utility cooperative games that involve a large set of players $1, \ldots, |N|$ and possibly depend on uncertain parameters. We first encode the game's utility function using a low-rank tensor decomposition, namely the tensor train (TT) model, which requires a limited number of function evaluations. The TT format casts the utility as a compressed tensor of shape $2^{|N|}$ and makes it possible to efficiently work with the exponentially-sized set of all possible coalitions of players. Given a game compressed in this manner, the proposed algorithm obtains arbitrary semivalues without incurring additional error, in particular the Shapley values and Banzhaf-Coleman indices, which are two of the most important allocation rules in cooperative game theory. Our algorithm takes $O(|N|R^2)$ operations per semivalue, where R is the game's TT rank. We show experimentally that many classical games can be compressed at low error with a moderate TT rank, making our algorithm more sample-efficient than Monte Carlo-based estimation. We also give a theoretical bound for the error of the semivalues obtained through our algorithm. Last, when the game depends on randomly distributed parameters, we are able to compute the expected semivalues efficiently.

Keywords: Cooperative game theory, semivalues, allocation rules, computational game theory, tensor approximation, tensor train decomposition

1. Introduction

Cooperative game theory studies the interactions between collaborative decision-makers (*players*) that are willing to pool their resources together in order to work towards a shared goal: completing a project, avoiding an expense, or more generally creating any form of value. A major question is how to measure each player's own merit at advancing that goal, since such merits can be used to determine fair player compensations (salaries, dividends, etc.).

A cooperative game is fully determined by the value $v(S)$ that each possible subset S of its players (or *coalition*) can produce. Examples of numerical value include the amount of units manufactured in an industrial process, money earned (or saved), voting power by parties in an assembly, etc. Clearly, the number of possible coalitions grows exponentially as $2^{|N|}$, where $|N|$ is the number of players. This fact can make it computationally prohibitive to work with large games and compute quantities of interest from them. For example, the R package `CoopGame` mentions an upper bound of around $|N| = 24$ players in modern commodity computers [1]. Moreover, these limitations are even tighter when the game depends on additional parameters. For this reason, it is often desirable to turn to alternative formulations or heuristics that can speed up such calculations under certain

assumptions. For instance, *generating functions* turn a game's utility function v into a polynomial, which is advantageous in weighted majority games with integer numbers of seats [2].

In this spirit, we propose to approximately encode v using a relatively small number of coefficients, namely via a *low-rank tensor train decomposition* (TT, for short), which is a particular case of *tensor network* [3]. Assuming that v admits a TT representation of rank at most R with acceptable accuracy, we contribute an algorithm that computes the game's *semivalues* [4] in a subexponential amount of operations and without incurring any additional error. To the best of our knowledge, ours is the first method to combine cooperative transferable utility games with low-rank tensor approximation. First we detail our algorithm for the simpler case of non-parameterized games (Sec. 3), and then describe its extension to games that are subject to uncertain parameters (Sec. 4). We find experimentally (Sec. 5) that a wide class of utility functions grow mildly in the TT format, i.e. for a constant accuracy, their maximal TT rank and number of compressed coefficients are asymptotically smaller than $O(2^{|N|})$. We also bound semivalue errors in terms of the approximation of v, and show empirically that they propagate mildly. We have released an open-source Python implementation of the proposed method[1].

Notation. Player sets are denoted using capital letters: N is the set of all players, while S is used for coalitions (subsets of players). The cardinality of a set S is written as $|S|$. Vectors and matrices use lowercase (resp. uppercase) bold letters, as in $\boldsymbol{i}$ and $\mathbf{I}$. We write a $|N|$-dimensional array (*tensor*) using a calligraphic letter as in $\mathcal{T}$, and index it in two ways:

- Either via a vector of indices in zero-based notation: $\mathcal{T}[\boldsymbol{i}] = \mathcal{T}[i_1, \ldots, i_{|N|}]$ where each i_k is either 0 or 1;
- Or via a player coalition: $\mathcal{T}[S] := \mathcal{T}[\boldsymbol{i}]$ where $i_k = 1$ if $k \in S$ and $i_k = 0$ otherwise. For instance, in a four-player game, $\mathcal{T}[\{1, 3\}]$ is a shortcut for $\mathcal{T}[1, 0, 1, 0]$.

As an error metric we use the *relative error* $\epsilon(\mathcal{T}^*, \mathcal{T}) := \|\mathcal{T} - \mathcal{T}^*\| / \|\mathcal{T}^*\|$, where $\mathcal{T}^*$ is a tensor, $\mathcal{T}$ is an approximation of $\mathcal{T}^*$, and $\|\cdot\|$ denotes the Frobenius norm of a tensor: $\|\mathcal{T}\| := \sum_{S \subseteq N} \mathcal{T}[S]^2$.

2. Related Work

2.1. Cooperative Games with Transferable Utility

Cooperative (sometimes also known as *coalitional*) game theory (CGT) is one of the two main branches of classical game theory [5]. Unlike non-cooperative games, players in CGT do not rely on a set of competing strategies, but have a shared objective and, before the game is carried out, they can fix their behavior by binding contracts known to all.

A cooperative game is completely determined by its *utility function*, which is a map $v : \{0, 1\}^{|N|} \to \mathbb{R}$; by convention, $v(\emptyset) = 0$. Since they operate on all subsets of a given set, utility functions may also be regarded as a hypergraph. If the game is *superadditive*, meaning that $v(S \cup T) \geq v(S) + v(T)$ for any pair of coalitions $S, T \subseteq N$, the game may also be interpreted in terms of set cardinalities (since $|N|$ sets can intersect in $2^{|N|}$ ways).

Throughout this paper we focus on transferable utility (TU) games only, meaning that any value obtained by virtue of player cooperation can be readily split among all participants. Under this

[1] `https://github.com/rballester/ttgames`.

assumption, the main question is what constitutes a fair distribution of profits: each player may interact differently with other players and, intuitively, his or her "merit" should depend on how much value they can add on average.

2.2. Semivalues

Semivalues are a framework for fair profit distribution: they split the *grand coalition* value $v(N)$ among all players by defining a payoff vector $\boldsymbol{\pi}$ [4], also known as *game solution*. Each player receives a weighted sum of his or her *marginal contributions*, namely how much value they can add to every possible coalition. Semivalues assign those weights based solely on each coalition's size $|S|$; more formally, the payoff allocated to player k is

$$\pi_k := \sum_{S \subseteq N \setminus \{k\}} p(|S|) \cdot (v(S \cup \{k\}) - v(S)) \tag{1}$$

where $v(S \cup \{k\}) - v(S)$ is player k's marginal contribution to coalition S and $p : \{0, \dots, |N|-1\} \to \mathbb{R}^{\geq 0}$ is a probability mass function (PMF) defined over the set of all coalitions of size $|N|-1$, i.e. satisfying $\sum_{s=0}^{|N|-1} \binom{|N|-1}{s} \cdot p(s) = \sum_{S \subseteq N \setminus \{k\}} p(|S|) = 1$. Note that the semivalues $\boldsymbol{\pi}$ are fully determined by the PMF $p(0), \dots, p(|N|-1)$.

Three of the most important semivalues include:

- The *Shapley value* [6], which we denote π^{Sh} and is defined by setting

$$p(|S|) := |S|!(|N| - |S| - 1)!/|N|! \tag{2}$$

  Shapley's solution has a number of attractive properties: it is *efficient*, *symmetric*, *additive*, and satisfies the *null player property* [7], and if the game is *convex*, it always belongs to its *core*. The applications of the Shapley value are wide-ranging and include estimating fair quotas in bankruptcy situations [8], ranking members of terrorist networks [9] and nodes in power grid networks [10], and more.

- The *Banzhaf-Coleman* semivalues or *Banzhaf power indices* [11] use $p(|S|) := 1/(2^{|N|-1})$. They are often applied to *weighted majority games*, where the value of a coalition is 1 if its members reach a certain number of seats in an assembly, and 0 otherwise. Importantly, Banzhaf indices measure the real ability of agents to make or break majorities in voting bodies [12, 13].

- The *binomial* semivalues [14] use $p(|S|) := \alpha^{|S|}(1-\alpha)^{|N|-|S|}$ where $\alpha \in [0, 1]$. They are useful when coalitions of a certain size (number of players) are more likely than others, and therefore are given more weight when accounting for possible partnership formations in e.g. political parliaments [15, 16].

2.3. Algorithms for Computing Semivalues

Semivalues are combinatorial in nature, which makes them challenging to compute: a naive, brute-force approach directly using Eq. 1 takes $O(2^{|N|})$ operations. In view of this, both exact and approximate methods to obtain π_k have been proposed in the literature. As an example of the former, Wang et al. [17] cast the problem as a matrix product by means of *semi-tensor products* (STP) [18], which are a useful tool to apply multilinear algebra techniques to multidimensional

arrays. Based on the STP, the authors give an exact and closed matrix formula to obtain any semivalue. Although compact, the formula still needs to evaluate the full set of $2^{|N|}$ coalitions and their values. Another exact approach is that of *generating functions* [2], which turns function v into a polynomial on the set of players N. It eliminates the exponential complexity under certain assumptions, but its main limitation is that it is only applicable to weighted majority games.

The second strategy, approximation, is typically based on unbiased Monte Carlo estimators; see [19] for estimation of the Shapley values and [12] for estimation of power indices in the particular case of simple games. The core idea is to use the following definition for the Shapley value, which traverses all permutations of N and is equivalent to Eq. 2:

$$\pi_k^{\mathrm{Sh}} = \frac{1}{|N|!} \sum_{\sigma \in \Pi(N)} (v(\sigma_k \cup \{k\}) - v(\sigma_k)), \tag{3}$$

where σ_k is the set of players that appear before k in permutation σ. Thanks to Eq. 3, one can estimate $\widehat{\boldsymbol{\pi}}^{\mathrm{Sh}} \approx \boldsymbol{\pi}^{\mathrm{Sh}}$ by generating random permutations of N, then incrementing the estimators $\widehat{\pi}_1^{\mathrm{Sh}}, \ldots \widehat{\pi}_{|N|}^{\mathrm{Sh}}$ for each permutation σ sampled [19]. This method has been extended to other allocation rules. For example, Saavedra-Nieves et al. [20] adapted it to the *Owen value*, which is a variant of the Shapley value with *a priori unions*, i.e. where coalitions must respect a given partition of N. More recently, Saavedra-Nieves and Fiestras-Janeiro [21] adapted the sampling method for the *Banzhaf-Owen value*, which applies a similar partition constrain to the Banzhaf value for simple games; it was applied to measure the voting power of members of the International Monetary Fund, which may only form coalitions within their respective constituencies. Although MC methods are very general and can often produce an acceptable estimation, note that the size $|N|!$ of the permutation space grows extremely quickly. This is even more challenging for the case of parameterized games, which have an even larger coalition space and our method can address (Sec. 4).

The algorithms proposed in this paper are a hybrid of these two approaches: although our semivalue formula is exact, it relies on first building a surrogate model (a tensor train, see next subsection) of the game's utility function that is subject to a variable approximation error. The main motivation for using compressed tensors is that a wide class of utility functions can be accurately represented as a low-rank expansion. Provided that such an approximation exists, the proposed method leads to accurate semivalue calculations using only a limited number of evaluations of v, more so than MC-based approaches. In Sec. 3.4 we will bound the error in the semivalues that results from the error in the approximation of v, and we give an experimental study in Sec. 6.

2.4. Tensor Train Model

Uncompressed tensors are a prime example of the *curse of dimensionality*, since their size grows exponentially with $|N|$. For this reason, low-rank decompositions have been successfully used to approximate tensors in a myriad of applications [22, 23]. In this context, the *tensor train* format [24] is a kind of approximator that is particularly effective for high-dimensional tensors, a property that we heavily exploit in this paper. Tensor trains mitigate the curse of dimensionality much better than the *Tucker decomposition* [25], especially for high numbers of variables. Moreover, the convergence properties of existing algorithms to learn a TT from a (possibly incomplete) tensor are better understood than those for the *CANDECOMP/PARAFAC* model [26]. In particular, we will make use of a heuristic known as *cross-approximation* (CA) that learns a tensor by visiting only a small fraction of its entries.

2.5. Tensors and Games

Classically, the *tensor decomposition* of a cooperative game [27] referred to its representation in terms of *Owen's composition* [28]. Other works use tensors to refer to multilinear (tensor-product) structure of utility functions in either cooperative [29] or non-cooperative games [30].

It is only recently that tensor approximation methods have been applied in computational game theory, and the combination of these two fields remains largely unexplored. [31] used a continuous analogue of the TT format to model the exponential number of states of a two-player pursuit-evasion game (which is non-cooperative). The method builds on a previous application of the TT format to the related field of optimal control [32]. In the context of uncertainty quantification, [33] extracted Shapley values (but not general semivalues) of a TT tensor containing the *Sobol indices* [34] of a response surface model. We are not aware of any existing work that uses low-rank tensor approximation to model general cooperative games.

3. Proposed Algorithm for Non-parameterized Games

Like most properties and quantities of interest of cooperative games, semivalues are challenging from a computational point of view: as per Eq. 1, each semivalue requires evaluating the game's utility function v over $2^{|N|-1}$ coalitions. Our algorithm computes π_k in three steps:

1. Form a low-rank TT tensor $\mathcal{T}$ of shape $2^{|N|}$ that approximately encodes $v(S)$ for every coalition $S \subseteq N$. This step is performed only once, as it is common to all semivalues π_k.
2. Out of $\mathcal{T}$, derive a TT tensor $\mathcal{T}_k$ that contains the marginal contribution $v(S \cup \{k\}) - v(S)$ of player k to every possible coalition S among players in $N \setminus \{k\}$. By construction, the TT rank of $\mathcal{T}_k$ is no larger than that of $\mathcal{T}$.
3. Sum all elements in $\mathcal{T}_k$, weighing each $S \subseteq N \setminus \{k\}$ by its corresponding semivalue probability $p(|S|)$. In turn, this weighted sum is computed as follows:
   (i) Form a weighting tensor $\mathcal{W}_k$ that contains $p(|S|)$ for all $S \subseteq N \setminus \{k\}$. The rank of $\mathcal{W}_k$ is fixed and equal to $|N| - 1$.
   (ii) Compute the dot product between tensors $\mathcal{T}_k$ and $\mathcal{W}_k$.

3.1. Approximating the Utility Function v as a TT Tensor $\mathcal{T}$

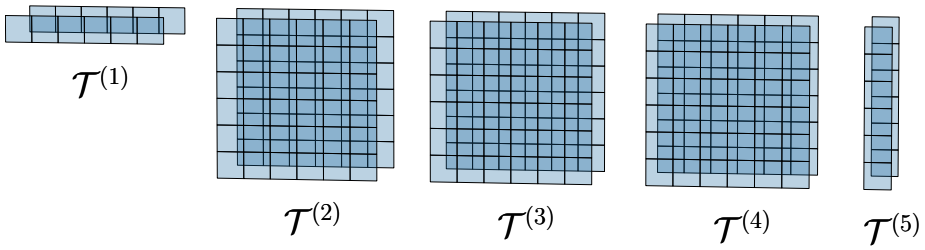
The TT format writes a tensor $\mathcal{T}$ as a sequence of three-dimensional tensors (*TT cores*), compactly represented as:

$$\mathcal{T} = [[\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(|N|)}]] \tag{4}$$

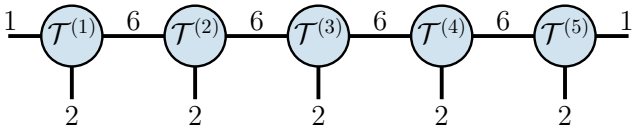
where each core $\mathcal{T}^{(k)}$ has shape $R_{k-1} \times I_k \times R_k$, $|N|$ is the tensor's dimensionality, and I_k is its size along dimension k. The vector $\boldsymbol{R}$ contains the so-called *TT ranks* and satisfy $R_0 = R_{|N|} = 1$. In our case, since the domain of v is $\{0,1\}^{|N|}$, each core has shape $R_{k-1} \times 2 \times R_k$ and can be regarded as a stack of two matrices: $\mathcal{T}^{(k)}[:,0,:]$ and $\mathcal{T}^{(k)}[:,1,:]$. The value of any coalition S is approximated as the product of the corresponding sequence of matrices:

$$v(S) \approx \mathcal{T}[S] = \mathcal{T}^{(1)}[:, i_1, :] \cdots \mathcal{T}^{(|N|)}[:, i_{|N|}, :] \tag{5}$$

where $i_k = 1$ if player k belongs to S and 0 otherwise. For instance, $v(N)$ is approximated by multiplying the second slice of all cores. See Fig. 1 for an illustration.



(a) Matrix-like representation



(b) Tensor network representation

Figure 1: A TT cooperative game with $|N| = 5$ players encodes the utility function v in five *TT cores*, each of which consists of two matrices. The value for every possible coalition $S \subseteq N$ is represented by the $2^{|N|} = 32$ possible matrix product sequences. In this example, the TT ranks are all equal to 6. In (b) we show each core as a node in a graph (a so-called tensor network).

3.1.1. Tensor Train Ranks

The $|N|-1$ TT ranks are defined as the matrix ranks of the input tensor's *unfoldings* $\mathbf{T}^{(1)}, \ldots, \mathbf{T}^{(|N|-1)}$, respectively [24]. In our case, the k-th unfolding matrix arises from grouping together the first k players to form the matrix's row index and the last $|N| - k$ players to form its column index. For example, if we model a three-player game $N = \{1, 2, 3\}$ as a utility tensor of shape 2^3, its two unfolding matrices will be:

$$\mathbf{T}^{(1)} = \begin{bmatrix} v(\emptyset) & v(\{2\}) & v(\{3\}) & v(\{2,3\}) \\ v(\{1\}) & v(\{1,2\}) & v(\{1,3\}) & v(\{1,2,3\}) \end{bmatrix}, \quad \mathbf{T}^{(2)} = \begin{bmatrix} v(\emptyset) & v(\{3\}) \\ v(\{1\}) & v(\{1,3\}) \\ v(\{2\}) & v(\{2,3\}) \\ v(\{1,2\}) & v(\{1,2,3\}) \end{bmatrix}. \tag{6}$$

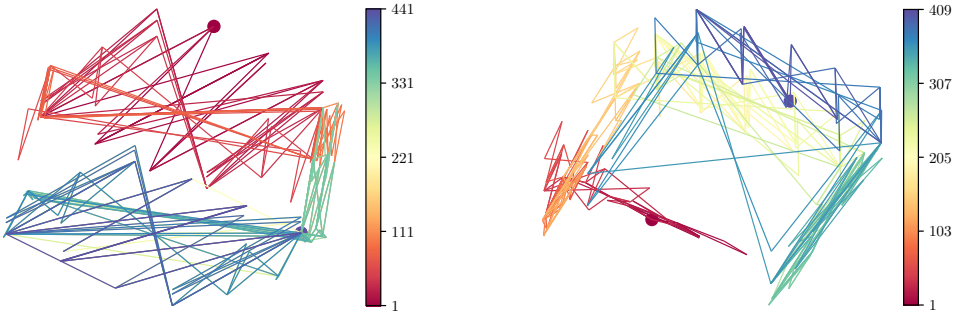
and its TT ranks R_1 and R_2 will be the matrix ranks of $\mathbf{T}^{(1)}$ and $\mathbf{T}^{(2)}$, respectively.

Note that the size of those unfolding matrices still depends exponentially on the number of players $|N|$ (each matrix $\mathbf{T}^{(k)}$ has shape $2^k \times 2^{|N|-k}$), but large space gains can be achieved if they are low rank. Even if that is not the case, the rank can often be truncated to a small number while introducing some approximation error. The rank choice is then the main trade-off of the format: the higher the rank, the higher the accuracy, but the storage and algorithmic costs will also increase accordingly. All in all, the format has storage cost $O(|N|R^2)$ where $R := \max\{R_1, \ldots, R_{|N|}\}$. As with every compression approach, there is of course *no free lunch*: in the worst case scenario, the unfolding matrices may have full rank and bring the total number of compressed TT coefficients to $O(2^{|N|})$, i.e. not better than the original, uncompressed utility function v. Fortunately, some games can be compressed exactly at low TT rank, and many others yield a manageable error at modest ranks that make the total storage roughly polynomial w.r.t. $|N|$.

3.1.2. Cross-approximation

In this paper, we derive the TT cores of a game without having to evaluate v over all possible coalitions. We do so by means of *cross-approximation* (CA) [35], a heuristic learning algorithm that designs a sampling plan on-the-fly seeking the best possible approximator $\mathcal{T}$ of $\mathcal{T}^*$ (in L^2 norm) with the fewest function evaluations. When used to learn a $2^{|N|}$ tensor, CA only needs to take $O(|N|R^2)$ samples from v, i.e. the number of evaluations is proportional to the number of coefficients of the resulting compressed tensor. The rank R is determined adaptively so as to reach a sufficiently small error $\epsilon(\mathcal{T}, \mathcal{T}^*) = \|\mathcal{T} - \mathcal{T}^*\|/\|\mathcal{T}^*\|$, according to a threshold given by the user. The samples are chosen iteratively in batches so as to best optimize one core at a time, with each iteration selecting appropriate rows and columns of one unfolding matrix by means of the *pseudoskeleton* approximation [36]. This procedure is partially stochastic, since the initial guesses for those rows and columns are chosen at random; see also [35, 37] for further discussion. These samples also serve as a validation set that helps estimate the error ϵ. At each iteration, the error drops as new ranks are added to the cores, and the iteration is stopped when it drops below a target threshold.

CA also relies on QR factorizations and matrix inversions of core slices, which are matrices of shape $O(R) \times O(R)$ and bring the algorithm's total computational cost to $O(|N|R^3)$ operations. As with any experimental design, there is again no free lunch: CA cannot *guarantee* any approximation accuracy unless all $2^{|N|}$ tensor entries are visited. In practice, however, it can often find a reasonable error after a few iterations. Fig. 2 illustrates the CA sampling plan for two games.



(a) 15-player *shoes* game (Sec. 5) (b) 14-player *many-sellers* game (Sec. 5)

Figure 2: First few hundred entries visited by the cross-approximation sampling scheme, colored by order and projected onto 2D via principal component analysis (PCA) of their coordinates. The traversal proceeds in a relatively structured way across the $2^{|N|}$ hypercube, progressively exploring different sets of coalitions.

3.2. Deriving the Marginal Contribution Tensor $\mathcal{T}_k$

Once $\mathcal{T}$ is available, we proceed to derive $\mathcal{T}_k$, a compressed tensor of shape $2^{k-1} \times 1 \times 2^{|N|-k}$ that encodes all possible contributions of player k to coalitions he does not belong to. We build $\mathcal{T}_k$ by leaving all cores untouched except for $\mathcal{T}^{(k)}$, which becomes the subtraction of its two slices; see Alg. 1.

Thee correctness of Alg. 1 follows from the linearity of matrix multiplication: let $\mathcal{T} \approx v$, let S be a coalition excluding player k, and consider the indices $i_1, \ldots, \widehat{i_k}, \ldots, i_{|N|}$ defined as $i_n = 1$ if $n \in S$ and 0 otherwise. Then, the tensor entry $\mathcal{T}_k[S]$ approximates the marginal contribution of k

Algorithm 1: Given a TT game containing the utility function values $v(S)$ for every possible coalition S among a set of players N, assemble another TT containing all possible marginal contributions for player k.

Input: Player k and TT cooperative game $\mathcal{T} = [[\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(|N|)}]]$
Result: A TT tensor
for $n = 1, \ldots, |N|$ **do**
  if $n = i$ **then**
    $\blacktriangleright\ \mathcal{T}_k^{(n)} := \mathcal{T}^{(n)}[:, 1, :] - \mathcal{T}^{(k)}[:, 0, :]$
  end
  else
    $\blacktriangleright\ \mathcal{T}_k^{(n)} := \mathcal{T}^{(n)}$
  end
end
return TT tensor $\mathcal{T}_k := [[\mathcal{T}_k^{(1)}, \ldots, \mathcal{T}_k^{(|N|)}]]$

to S:

$$\begin{aligned}\mathcal{T}_k[S] = \mathcal{T}_k[i_1, \ldots, i_{k-1}, 0, i_{k+1}, \ldots, i_{|N|}] = \\ = \left(\prod_{n=1}^{k-1} \mathcal{T}^{(n)}[:, i_n, :]\right) \cdot \left(\mathcal{T}^{(k)}[:, 1, :] - \mathcal{T}^{(k)}[:, 0, :]\right) \cdot \left(\prod_{n=k+1}^{|N|} \mathcal{T}^{(n)}[:, i_n, :]\right) = \\ = \mathcal{T}[i_1, \ldots, i_{k-1}, 1, i_{k+1}, \ldots, i_{|N|}] - \mathcal{T}[i_1, \ldots, i_{k-1}, 0, i_{k+1}, \ldots, i_{|N|}] \approx v(S \cup \{k\}) - v(S).\end{aligned} \tag{7}$$

Note that Alg. 1 does not increase the size of any TT core. In particular, all TT ranks of $\mathcal{T}_k$ are the same as they were in the original $\mathcal{T}$.

3.3. Computing the Weighted Sum

The third step to obtain semivalue π_k is to sum every entry S of tensor $\mathcal{T}_k$ weighted by $p(|S|)$, where p is the semivalue's PMF (recall Sec. 2.2). We cast this as a *tensor dot product* operation:

$$\pi_k = \langle \mathcal{T}_k, \mathcal{W}_k \rangle , \tag{8}$$

where $\mathcal{W}_k$ is a suitable *weighting tensor* that contains each coalition's probability as per the semivalue of interest, i.e. $\mathcal{W}_k[S] = p(|S|)$ or, equivalently, $\mathcal{W}_k[\boldsymbol{i}] = p(\sum_{n=1}^{|N|} i_n)$.

3.3.1. Assembling the Weighting Tensor $\mathcal{W}_k$

Note that the first matrix in Eq. 5 is a row vector; when evaluated from left to right, it is actually a sequence of vector-matrix products: $w_{n+1} = \mathcal{T}^{(n)}[:, i_n, :] \cdot w_n$ with $w_1 = \mathcal{T}^{(1)}[:, i_1, :]$. The idea to build $\mathcal{W}_k$ is to handcraft its TT tensor cores $\mathcal{W}_k^{(1)}, \ldots, \mathcal{W}_k^{(|N|)}$ so that, when evaluating $\mathcal{W}_k[S]$, each partial result w_n, $1 \leq n \leq |N|$, satisfies the following:

$$\begin{cases} w_n[i] = 1 \text{ if } |S \cap \{1, \ldots n\}| = i \\ w_n[i] = 0 \text{ otherwise} \end{cases} \tag{9}$$

In other words, each w_n uses *one-hot encoding* to count how many players belong to S as we progress from player 1 to $|N|$. To accomplish Eq. 9, we assemble each core by stacking an identity matrix and an identity matrix shifted towards the right; the former applies when $n \notin S$ and does not increment the counter (i.e. $w_{n+1} = \mathbf{I} \cdot w_n = w_n$), while the latter applies when $n \in S$ and increments the counter by 1 (i.e., shifts w_n one position towards the right). The only two exceptions are:

- Core $\mathcal{W}_k^{(k)}$ does not have a second slice, since we are only considering coalitions k does not belong to;
- Core $\mathcal{W}_k^{(|N|)}$ is used to hold the semivalue probabilities $p(0) \dots p(|N|-1)$. Since the last matrix in Eq. 5 is a column vector, the last product is a vector-vector dot product whose result is the desired scalar $w_{|N|+1} = p(|S|)$.

Since coalition S can contain up to $|N|-1$ players, a TT rank of $R = |N|-1$ is enough to encode all possible values of the counter. See Alg. 2 for full details, and Fig. 3 for an example illustration of a five-player game. The storage cost of $\mathcal{W}_k$ is $O(|N|^3)$ coefficients.

Algorithm 2: Assemble a weighting tensor that weighs every possible coalition S among players $N \setminus \{k\}$ by its probability $p(|S|)$.

Input: Player k, semivalue function $p : \{0, \dots, |N|-1\} \to \mathbb{R}^{\geq 0}$
Result: An $|N|$-dimensional tensor $\mathcal{W}_k$ of shape $2^{i-1} \times 1 \times 2^{|N|-i-1}$
for $n = 1, \dots, |N|-1$ **do**
  ▶ $\mathcal{W}_k^{(n)}[:, 0, :] := \mathbf{I}_{(|N|-1)\times(|N|-1)}$ // Identity matrix
  ▶ $\mathcal{W}_k^{(n)}[:, 1, :] := \begin{bmatrix} \mathbf{0} & | & \mathbf{I}_{(|N|-1)\times(|N|-2)} \end{bmatrix}$ // Identity matrix, shifted one position to the right
end
▶ $\mathcal{W}_k^{(|N|)} := \text{zeros}((|N|-1) \times 2 \times 1)$
for $s = 0, \dots, |N|-1$ **do**
  ▶ $\mathcal{W}_k^{(|N|)}[s, 0, 0] := p(s)$
  ▶ $\mathcal{W}_k^{(|N|)}[s, 1, 0] := p(s+1)$
end
▶ $\mathcal{W}_k^{(k)} = \mathcal{W}_k^{(k)}[:, 0, :]$
return TT tensor $\mathcal{W}_k := [[\mathcal{W}_k^{(1)}, \dots, \mathcal{W}_k^{(|N|)}]]$

3.3.2. Tensor Dot Product

Once we have assembled both $\mathcal{T}_k$ and $\mathcal{W}_k$ as TT tensors, the last step is to find the dot product between them (Eq. 8). We do this directly in the TT compressed domain via a well-known procedure that is due to [24] and relies on a sequence of $2|N|-1$ tensor-tensor contractions; see also [38]. For two tensors of shape $I^{|N|}$ and TT rank R and S, the dot product's time cost is $O(|N|I(R^2S + RS^2))$. In our case, which uses two $(2^{|N|-1})$-shaped tensors of ranks R and $|N|-1$, this algorithm amounts to $O(|N|^2R^2 + |N|^3R)$ operations. In practice, however, since $\mathcal{W}_k$ is highly sparse and mostly

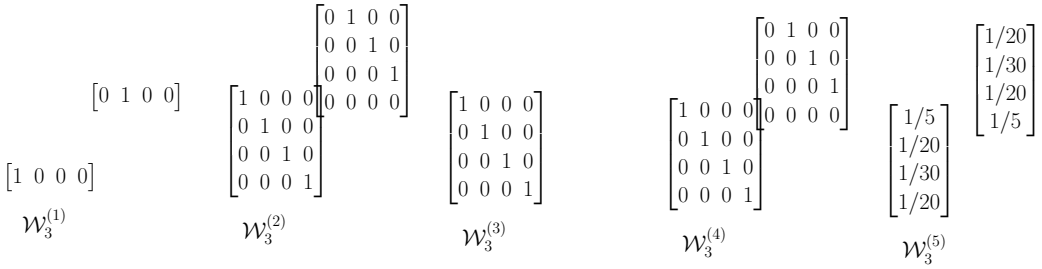


Figure 3: Example of weighting tensor $\mathcal{W}_3$ encoding all marginal contributions of player 3 as per Alg. 2. In this five-player game, $\mathcal{W}_3$ is represented in the TT format using five cores $\mathcal{W}_3^{(1)}, \ldots, \mathcal{W}_3^{(5)}$, has TT ranks $|N| - 1 = 4$, and satisfies $\mathcal{W}_3[S] = p(|S|)$ for every $S \subseteq N \setminus \{3\}$. Here we use the binomial semivalue with $\alpha = 0.5$, whose vector of probabilities is $\boldsymbol{p} = (1/5, 1/20, 1/30, 1/20, 1/5)$.

consists of variations of identity matrices, the final cost we incur to obtain semivalue π_k is simply $O(|N|R^2)$.

3.4. Error Bound

Unlike MC-based estimators [19, 12] that use a subset of the exact entries of v, the proposed method can work with full set of coalitions and their values thanks to the TT representation $\mathcal{T}$; however, those values are subject to an approximation error $\mathcal{T} - \mathcal{T}^*$, where $\mathcal{T}^*$ denotes an error-free tensor representing the true coalition values v. Next, we work out an error bound for the proposed semivalue computation.

Proposition 1. *Let π_k^* denote the true semivalue we want to approximate. Then,*

$$|\pi_k - \pi_k^*| \leq \|\mathcal{T} - \mathcal{T}^*\| \cdot \sqrt{\sum_{n=0}^{|N|-1} \left[\binom{|N|-1}{n} p(n)^2 \right]} \tag{10}$$

Proof. Let $\mathcal{T}_k^*$ be the marginal contribution tensor of $\mathcal{T}^*$ for player k. Then, by the Cauchy-Schwarz inequality,

$$|\pi_k - \pi_k^*| = |\, \langle \mathcal{T}_k, \mathcal{W}_k \rangle - \langle \mathcal{T}_k^*, \mathcal{W}_k \rangle \,| = |\, \langle \mathcal{T}_k - \mathcal{T}_k^*, \mathcal{W}_k \rangle \,| \leq \|\mathcal{T}_k - \mathcal{T}_k^*\| \|\mathcal{W}_k\|. \tag{11}$$

Next, we bound $\|\mathcal{T}_k - \mathcal{T}_k^*\|$ by noting that $\mathcal{T}_k = \mathcal{T}[1] - \mathcal{T}[0]$ where $\mathcal{T}[0]$ and $\mathcal{T}[1]$ denote the first and second slices of $\mathcal{T}$ along dimension k, respectively (Sec. 3.2). Therefore,

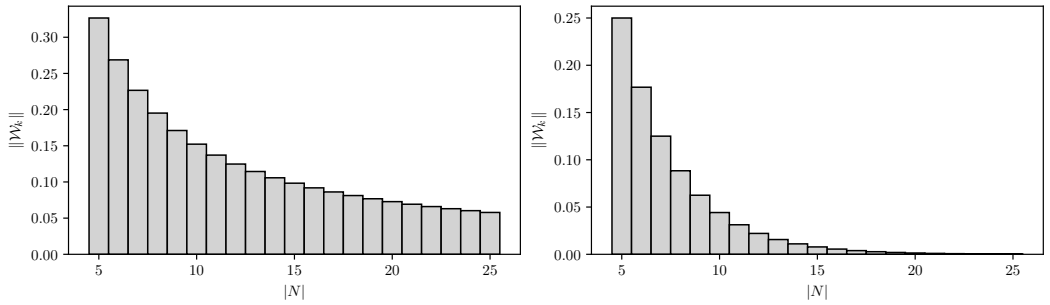
$$\begin{aligned} \|\mathcal{T}_k - \mathcal{T}_k^*\|^2 = \|(\mathcal{T}[1] - \mathcal{T}[0]) - (\mathcal{T}^*[1] - \mathcal{T}^*[0])\|^2 = \\ \|(\mathcal{T}[1] - \mathcal{T}^*[1]) - (\mathcal{T}[0] - \mathcal{T}^*[0])\|^2 \leq \\ \left(\|\mathcal{T}[1] - \mathcal{T}^*[1]\| + \|\mathcal{T}[0] - \mathcal{T}^*[0]\|\right)^2 \text{ (by the triangular inequality)} \leq \\ \|\mathcal{T}[1] - \mathcal{T}^*[1]\|^2 + \|\mathcal{T}[0] - \mathcal{T}^*[0]\|^2 = \|\mathcal{T} - \mathcal{T}^*\|^2 \end{aligned} \tag{12}$$

To conclude the proof, note that the sum of squared elements of $\mathcal{W}_k$ equals

$$\|\mathcal{W}_k\|^2 = \sum_{n=0}^{|N|-1} \binom{|N|-1}{n} p(n)^2, \tag{13}$$

since each $\mathcal{W}_k$ only contains the values $p(0), \ldots, p(|N|-1)$ and each value $p(n)$ appears exactly $\binom{|N|-1}{n}$ times. □

To give an idea of the magnitude of $\|\mathcal{W}_k\|$, we plot it in Fig. 4 for the Shapley and Banzhaf-Coleman semivalues over a range of $|N|$. In practice, the actual error is well below the theoretical bound of Eq. 10; we do a comparison in Sec. 6.



(a) Shapley (b) Banzhaf-Coleman

Figure 4: The weighting tensor $\mathcal{W}_k$ has a moderate norm that decreases with $|N|$. Note that the choice of k has no influence.

4. Extension to Parameterized Games

Thanks to the TT format being a powerful tool to circumvent the curse of dimensionality, we can also efficiently find semivalues of *parameterized* games, i.e. those whose utility function $v(S;\boldsymbol{\theta})$ depends on a parameter vector $\boldsymbol{\theta} = \theta_1, \ldots, \theta_M$. Note that, if done via direct application of the formula in Sec. 3, obtaining semivalues over a whole parameter space is exponentially more computationally demanding than when the parameters are fixed.

To do this efficiently in the TT compressed domain, we first discretize the parameter space $\boldsymbol{\theta}$ so that each θ_k can only take a finite set of values $\{\theta_k(1), \ldots, \theta_k(I)\}$. Thus, the function $v(S;\boldsymbol{\theta})$ can be regarded as a matrix $\mathbf{M}$ of shape $I^M \times 2^{|N|}$. In our experiments (Sec. 6.3) we work with $M = |N|$ parameters. We organize $\mathbf{M}$ as a tensor of shape $(2I)^{|N|}$ that we encode using the so-called *tensor train matrix* (TTM) [39], also known as *matrix product operator*. The TTM format adds one dimension to each core, which in our case becomes $(R_{k-1} \times I \times 2 \times R_k)$-shaped; the core's second dimension indexes the $I^{|N|}$ parameter combinations (the matrix rows), while its third dimension indexes the $2^{|N|}$ coalitions (the matrix columns). We denote this TTM matrix with the letter $\mathcal{M}$. See Fig. 5 for an illustration.

Once we have $\mathcal{M}$, we computing player k's marginal contributions $\mathcal{M}_k$ analogously to the non-parameterized case (Alg. 1), with the difference that, this time, we subtract the slices along the third dimension. Then, computing a player's semivalue for the entire set of parameters is compactly written as

$$\mathcal{P}_k = \mathcal{M}_k \cdot \mathcal{W}_k, \tag{14}$$

where $\cdot$ denotes a matrix-vector product. In the compressed domain, this product can be done one

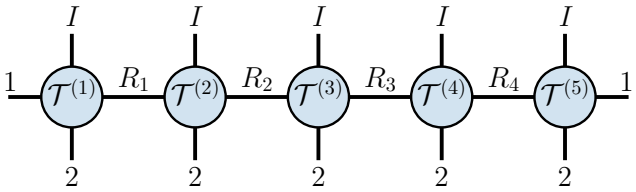


Figure 5: A TT matrix (TTM) represents a $I^{|N|} \times J^{|N|}$ matrix as a tensor of shape $(IJ)^{|N|}$ that is compressed as a sequence of $|N|$ 4D tensor cores. When used to compress a parameterized cooperative game, each core $\mathcal{T}^{(k)}$ has shape $R_{k-1} \times I \times 2 \times R_k$.

core slice at a time as follows:

$$\mathcal{P}^{(n)}[:, i, :] := \sum_{j=0}^{j=1} \left(\mathcal{M}_k^{(n)}[:, i, j, :] \otimes \mathcal{W}_k^{(n)}[:, j, :] \right), 0 \leq i \leq I-1, n = 1, \ldots, |N|. \tag{15}$$

Eq. 15 takes a total of $O(|N| R_1^2 R_2^2 I)$ operations, where R_1 and R_2 are the ranks of $\mathcal{M}_k^{(n)}$ and $\mathcal{W}_k^{(n)}$, respectively. The resulting $\mathcal{P}_k$ is a TT tensor of shape $I^{|N|}$, which contains the k-th semivalue for all parameter combinations in the discretized domain $\boldsymbol{\Theta}$.

Once the parameterized semivalues are available as a TT vector $\mathcal{P}_k$, one can compute other quantities of interest. For example, if the vector of input parameters is a random variable with known joint probability density function (PDF), one may obtain *expected Shapley values*. If the PDF is discretized over the same grid and approximated as a TT tensor $\mathcal{D}$, then we have

$$\mathbb{E}[\mathcal{P}_k] = \langle \mathcal{P}_k, \mathcal{D} \rangle, \tag{16}$$

which is a well-known formula to find the expectation of a TT tensor (such multidimensional integration operations were the main drivers behind the development of the TT format in the first place; see [24]).

In the important particular case where the parameters are independently distributed with discretized marginal distributions $\boldsymbol{m}^{(1)}, \ldots, \boldsymbol{m}^{(|N|)}$, then the expected semivalue for player k is given by

$$\mathbb{E}[\mathcal{P}_k] = \left\langle \mathcal{P}_k, \boldsymbol{m}^{(1)} \otimes \cdots \otimes \boldsymbol{m}^{(|N|)} \right\rangle = \mathcal{P}_k \times_1 \boldsymbol{m}^{(1)} \times_2 \cdots \times_{|N|} \boldsymbol{m}^{(N)}, \tag{17}$$

where $\times_n$ denotes the *tensor-times-vector product* along the n-th mode [3]. In the TT format, if the cores of $\mathcal{P}_k$ are denoted $\mathcal{P}_k^{(1)}, \ldots, \mathcal{P}_k^{|N|}$ and have rank at most $\widehat{R}$, Eq. 17 can be computed in just $O(|N| \widehat{R}^2)$ operations via the following sequence of matrix products:

$$\mathbb{E}[\mathcal{P}_k] = \left(\sum_{i=0}^{I-1} \mathcal{P}_k^{(1)}[:, i, :] \cdot \boldsymbol{m}^{(1)}[i] \right) \cdots \left(\sum_{i=0}^{I-1} \mathcal{P}_k^{(|N|)}[:, i, :] \cdot \boldsymbol{m}^{(|N|)}[i] \right). \tag{18}$$

5. Simulation Models

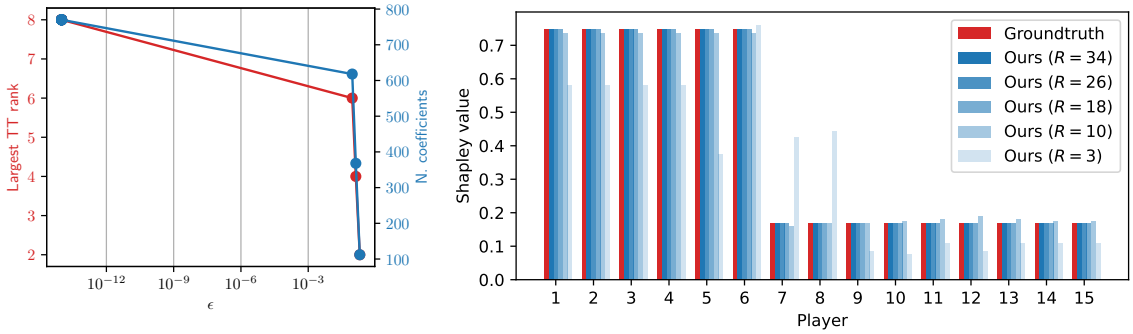
Next, we demonstrate the performance of the proposed algorithm in six classical cooperative games. In each game, the relative error $\epsilon(v, \mathcal{T}) = \|v - \mathcal{T}\| / \|v\|$ incurred by the cross-approximation step (Sec. 3.1) was estimated using a validation set of 10'000 samples distributed at random among

the set of possible coalitions. Groundtruth semivalues were found as per Eq. 1 by iterating over all coalitions. All experiments were conducted in Python 3.6 using *NumPy*[2] for general numerical routines and *tntorch*[3] for tensor manipulation routines. We used a 4-core i5-6600 3.3GHz Intel workstation with 16GB RAM; no GPU parallelism was exploited.

Shoe Sale

In this well-known cooperative game, players $1, \dots, L$ own left shoes while players $L+1, \dots, 2L+1$ own right shoes. All pairs belong to the same shoe model and size, so every left shoe matches every right shoe. Shoes can only be sold in pairs, hence the value of a coalition is the number of pairs they can form together.

We set $L := 7$ for a total of $|N| = 15$ players. In Fig. 6 we gradually increase the TT rank budget for CA and show the accuracy achieved in each case, both in terms of the approximated utility function v and the resulting Shapley values. We see that a rank of $R = L$ is enough to achieve round-off precision, and the Shapley values are accordingly accurate.



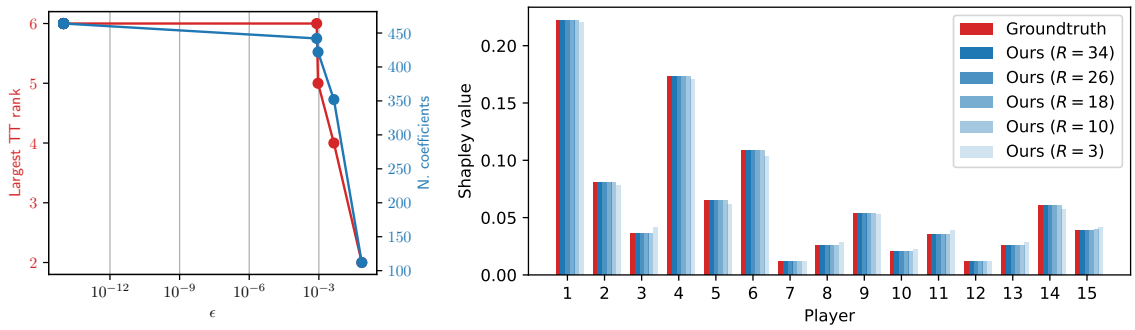
(a) (b)

Figure 6: *Shoes* game. Left: relative error of v and number of TT coefficients attained at various rank budgets. Right: Shapley values at different ranks compared against the groundtruth.

Airport

In this example, originally proposed by [40], each player $1 \leq k \leq |N|$ is an aircraft operator that needs a runway at least c_k meters long. This is a *cost* game, not a payoff one, but all formulations and mathematical treatment are analogous. The cost of a coalition is $v(S) = \max\{c_k | k \in S\}$, where $\boldsymbol{c}$ is a vector representing the runway building cost required by each company. Fig. 7 shows empirical accuracy results for $|N| = 15$ operators where each constant c_k is selected uniformly at random in $[0, 1]$. Rank $R = 4$ and above was sufficient to produce accurate semivalues.

[2] `https://www.numpy.org`.
[3] `https://github.com/rballester/tntorch`.

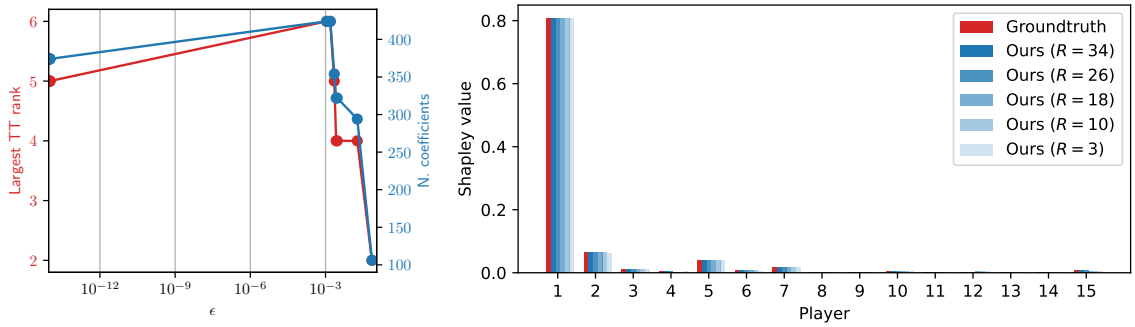


(a) (b)

Figure 7: *Airport* game. Left: relative error of v and number of TT coefficients attained at various rank budgets. Right: Shapley values at different ranks compared against the groundtruth.

One Seller Market

Next, we test a version of an auction game in the spirit of [41]. Player 1 is selling an object of no intrinsic worth to himself, while players $2, \ldots, |N|$ are interested potential buyers. Each buyer k values the object at $a_k > 0$ dollars. If the seller belongs to a coalition S, its value is that of the highest participating bidder: $v(S) = \max_{k \in S}\{a_k\}$. Conversely, if the seller does not belong to S, then $v(S) = 0$ because no trade can occur. We pick $|N| = 15$ again and set each valuation at random in the interval $[0, 1]$. As can be seen in Fig. 8, a rank of $R = 5$ was sufficient to attain round-off precision.



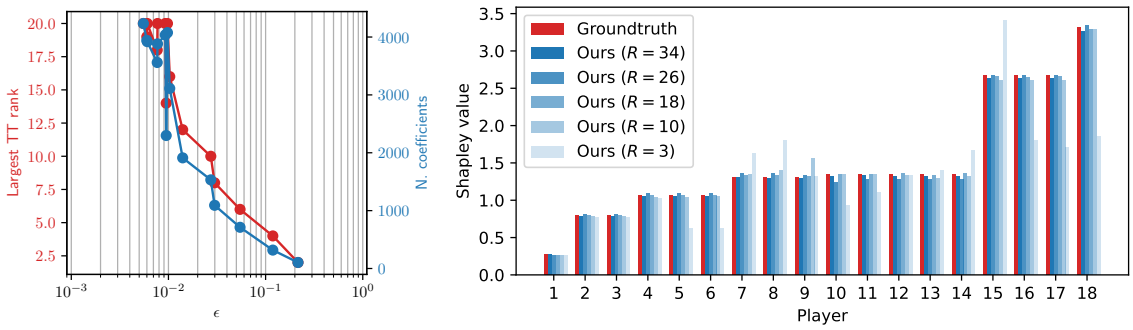
(a) (b)

Figure 8: *One seller market* game. Left: relative error of v and number of TT coefficients attained at various rank budgets. Right: Shapley values at different ranks compared against the groundtruth.

Many Sellers Market

This is a variation of the previous game in which there is a set of buyers B and a set of sellers C offering a divisible commodity. The former ask prices $a_1, \ldots, a_{|B|}$, while the latter offer prices $a_{|B|+1}, \ldots, a_{|B|+|C|}$. The value of a coalition is that of the total possible trade among its

participants: $v(S) = \min\{\sum_{k\in B} a_k, \sum_{k\in C} a_k\}$. We pick $|B| = |C| = 7$ (hence, $|N| = 14$), and draw the valuations a_k as random integers in $[1, 10]$. See Fig. 9 for the empirical results.

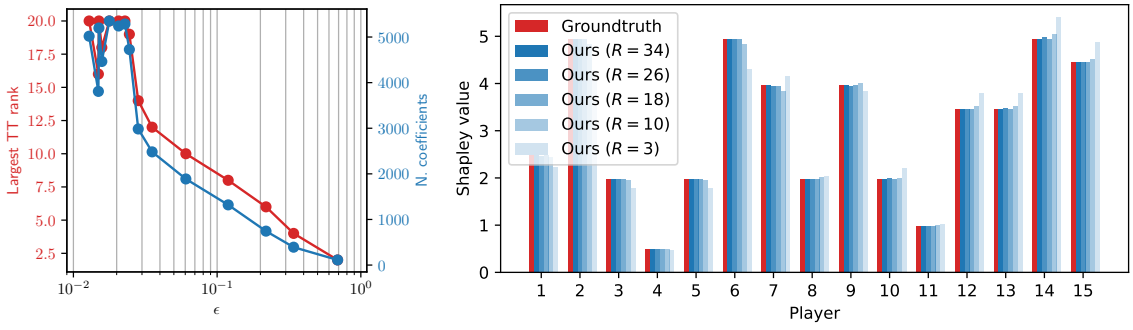


(a) (b)

Figure 9: *Many sellers market* game. Left: relative error of v and number of TT coefficients attained at various rank budgets. Right: Shapley values at different ranks compared against the groundtruth.

Bankruptcy

This classic game was already proposed in the Talmud. It considers $|N|$ creditors that have claims $c_1, \ldots, c_{|N|}$ over an estate whose value E is less than the sum of all claims [42]. The value of a coalition S is defined as $\max\{0, E - \sum_{k\notin S} c_k\}$. We pick $|N| = 15$, draw each claim c_k as a random integer between 1 and 10, and define $E := \frac{1}{2} \cdot \sum_k c_k$. We report accuracy results in Fig. 10



(a) (b)

Figure 10: *Bankruptcy* game. Left: relative error of v and number of TT coefficients attained at various rank budgets. Right: Shapley values at different ranks compared against the groundtruth.

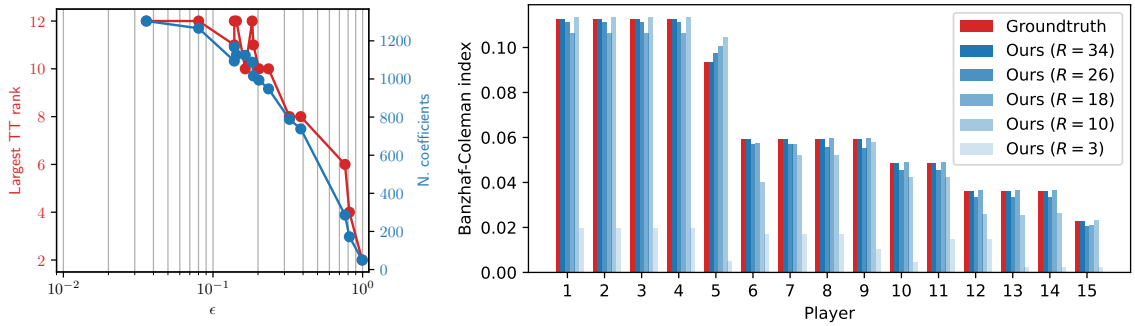
Weighted Majority

Weighted majority games are a prominent case of the so-called *simple* games, in which each player has a certain voting power w_k and a coalition S can lead to two outcomes only:

$$\begin{cases} v(S) = 1 \text{ if } \sum_{k \in S} w_k \geq M \text{ (i.e. } S \text{ reaches a majority of votes)} \\ v(S) = 0 \text{ otherwise } (S \text{ fails to reach majority} \end{cases} \tag{19}$$

In this scenario, the Banzhaf-Coleman semivalues (recall Sec. 2.2) enjoy a particularly interpretable application: they count how many times a player is critical for a coalition to reach majority.

We consider the game $\boldsymbol{w} = (10, 10, 10, 10, 8, 5, 5, 5, 5, 4, 4, 3, 3, 3, 2)$ with $M = 62$ which, prior to 2004, was the seat distribution among the 15 countries in the European Union's Council of Ministers [43], and we report the tensor approximation error and calculated Banzhaf-Coleman indices in Fig. 11.



(a) (b)

Figure 11: *Weighted majority* game. Left: relative error of v and number of TT coefficients attained at various rank budgets. Right: Banzhaf-Coleman values at different ranks compared against the groundtruth.

6. Further Analysis

6.1. Error Analysis

By building an approximate low-rank expansion of each game, the proposed method outperforms Monte Carlo estimation [19] and achieves lower absolute semivalue errors for an equivalent number of samples; see Tab. 1 for a comparison demonstrating this over all six games considered and $|N| = 20$. In the case of *Weighted Majority*, the number of seats of each party was chosen as a random integer in $[1, 10]$, and the threshold is a simple majority rule: $M := \lfloor \frac{1}{2} \cdot \sum_k w_k \rfloor + 1$. We also observe those errors to be well below the upper bound we provided in Sec. 3.4. Last, note that relative error estimations undertaken during cross-approximation are close to actual

6.2. Time Measurements

Fig. 12 reports the running times for our method as compared to computing semivalues via Eq. 1 (brute-force enumeration of all coalitions). The number of players is varied from 14 to 22, both included, in order to reflect the scalability of each algorithm. The CA iteration was stopped

	Metric	Shoes	Airport	O. Seller	M. Sellers	Bankruptcy	W. Majority
Ours	Rel. error (CA)	0.0028	0.015	0.0034	0.013	0.031	0.21
	Rel. error (actual)	0.003	0.016	0.0028	0.015	0.03	0.21
	Abs. error (actual)	13.21	14.73	1.85	392.16	321.33	153.41
	Error bound (Sec. 3.4)	0.96	1.07	0.13	28.55	23.39	11.17
	Avg. $\lvert\pi_k - \pi_k^*\rvert$	0.0016	0.0017	$9.44 \cdot 10^{-5}$	0.039	0.0034	0.0015
	N. samples	13644	3272	4908	18228	19916	41196
MC [19]	Avg. $\lvert\pi_k - \pi_k^*\rvert$	0.015	0.0062	0.0025	0.053	0.083	0.0035
	N. samples	13640	3260	4900	18220	19900	41180

Table 1: Comparison between our method and Monte Carlo estimation of the Shapley values. Note that for a similar number of samples, the proposed method leads to a lower mean absolute error $|\pi_k - \pi_k^*|$. The ground-truth was obtained using brute-force, and the number of players is $|N| = 20$ in all cases for a total of $2^{10} \approx 10^6$ possible coalitions. The rows *absolute error* and *relative error* refer to the full tensor of coalitions, i.e. $\|\mathcal{T} - \mathcal{T}^*\|$ and $\|\mathcal{T} - \mathcal{T}^*\|/\|\mathcal{T}^*\|$, respectively.

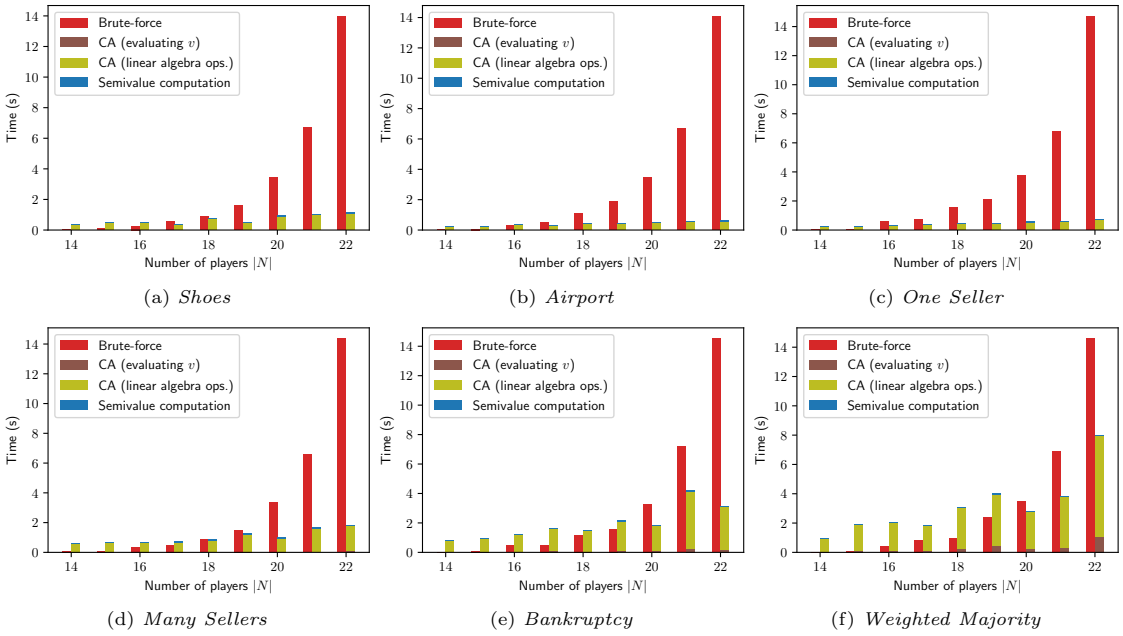
as soon as the error threshold $\epsilon = 10^{-2}$ was reached. All parameters in games 5 to 5 were drawn from the same distributions as in previous experiments. We observe an exponential time cost w.r.t. $|N|$ for the brute-force method in all cases, in contrast to the algorithms proposed in Sec. 3 which take sub-exponential time. Moreover, the proposed dot product step (Secs. 3.2 and 3.3) take a very small amount of time compared to learning the TT tensor in the first place (Sec. 3.1).

6.3. Parameterized Utility Functions

As a last experiment, we learn tensors that approximate parameterized games, then compute their expected semivalues as per Sec. 4.

We consider first the one-seller game (Sec. 5) when the buyer bids are not constant but independently distributed random variables: $a_k \sim \mathcal{U}(0,1), 2 \leq k \leq K$. We pick $|N| = 6$ and discretize each interval $[0,1]$ into $I = 32$ values $\{0, \frac{1}{31}, \ldots, 1\}$. To make $M = |N|$ as in Sec. 4, we prepend a dummy parameter a_1 that has no effect and can only take one value. To form a TT matrix, we attach each parameter to the TT core of its corresponding player to form a 4D core, is left unchanged. All in all, the resulting parameterized game is a tensor $\mathcal{T}$ of shape $2^6 \cdot 32^5 \approx 2.14 \cdot 10^9$ elements. Note that a brute-force computation of the semivalues over all parameter combinations would require a number of operations of the same magnitude. CA took $6.42 \cdot 10^5$ evaluations of $v(s, \boldsymbol{\theta})$ to build $\mathcal{T}$ and attained a relative error $\epsilon(v, \mathcal{T}) \approx 7.37 \cdot 10^{-3}$ (as usual, estimated using a 10'000-sized validation set). CA took 0.66s (0.20s for utility function evaluations plus 0.46s for linear algebra operations), while extracting the parameterized semivalue tensors $\mathcal{P}_1$ and $\mathcal{P}_2$ took 0.13s on average. Fig. 13 shows how different buyer bids influence two of the expected semivalues.

Second, we study the *Bankruptcy* game (Sec. 5) when the estate E and the creditors' claims follow $\mathcal{U}(0,2)$ and $\mathcal{U}(0,1)$, respectively, for a total of $M = 7$ parameters. We pick $M = 7$ and, in order to make $M = |N|$, we prepend a dummy player 0 with no effect. This results in a tensor of size $2^6 \cdot 32^7 \approx 2.20 \cdot 10^{12}$ elements. CA took approximately $9.79 \cdot 10^6$ function evaluations to learn this tensor and resulted in $\epsilon \approx 6.99 \cdot 10^{-3}$. CA took 5.32s (2.94s for utility function evaluations plus 2.39s for linear algebra operations), while extracting the parameterized semivalue tensor $\mathcal{P}_1$ took 2.09s. Fig. 14 shows one of the expected semivalues as a function of two of the parameters.



(a) *Shoes* (b) *Airport* (c) *One Seller*

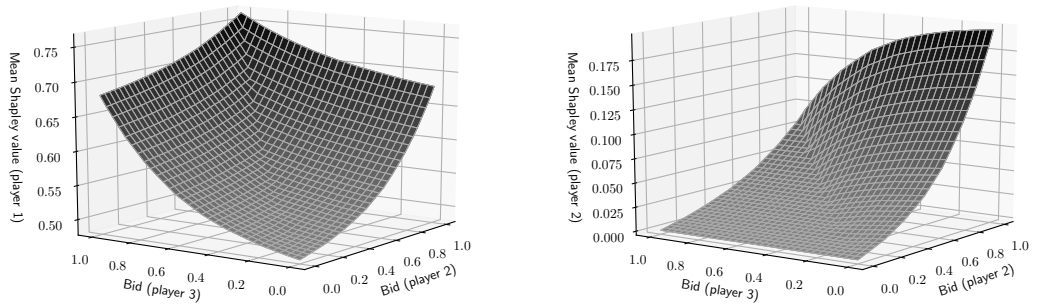
(d) *Many Sellers* (e) *Bankruptcy* (f) *Weighted Majority*

Figure 12: Time needed by the proposed algorithm at constant error threshold $\epsilon = 10^{-2}$ and increasing number of players. The brute-force approach uses Eq. 1 directly and, due to its exponential blow-up, it quickly becomes increasingly impractical beyond $|N| = 20$ players.

7. Conclusions

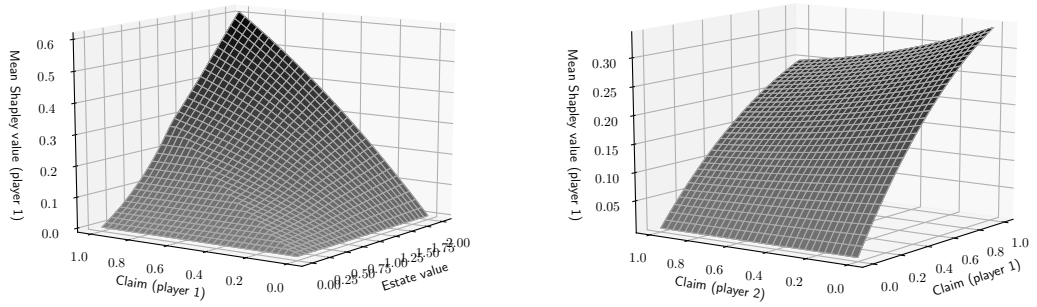
We gave an algorithm that transforms the semivalue computation task into a tensor dot product, which can be evaluated efficiently using tensor contractions. Our idea relies first on casting the utility function of a cooperative game as an $|N|$-dimensional tensor train, where N is the set of players. The TT format grows in size only quadratically with respect to the target tensor's rank and, depending on the utility function's structure, this representation can lead to exponential space and time savings at the cost of a variable approximation error. In order to build the compressed tensor without visiting all $2^{|N|}$ entries of the utility function, we use cross-approximation, an adaptive sampling algorithm that learns a low-rank TT tensor from any black-box function. Our semivalue algorithm is exact on the TT tensor, i.e. the whole procedure only may introduce errors during the initial tensor learning stage. We also extended this idea to parameterized games, whereby we are able to obtain parameterized Shapley values and compute expected semivalues. Examining those, one can gain insights on how any player's "merit" or "fair share" varies as parameters are moved.

In summary, we showed that the proposed method can effectively estimate semivalues using a moderate number of samples as long as the utility function of interest exhibits low-rank structure. As a limitation of our method, note that there is no free lunch and this low-rank assumption will not always hold. Random sampling remains the most general approach since it is applicable to arbitrary utility functions and numbers of players without underlying assumptions. In contrast, cross-approximation has cost $O(|N|R^2)$ that is quadratic on the rank R and is usually limited to moderately high dimensionalities (say, $|N| \leq 100$). In other words, very large games involving hundreds or thousands of players are currently out of reach for this method.



(a) Seller (b) First buyer

Figure 13: Parameterized Shapley values for the *one seller* game (Sec. 5), computed as per Sec. 4. Depicted are the expected Shapley values for the seller (left) and first buyer (right) as a function of two of the bids. All bids are uniformly distributed in $[0, 1]$.



(a) (b)

Figure 14: Expected Shapley value for the first creditor in a six-player *Bankruptcy* game as a function of the creditor's claim and the estate value (left) vs. another creditor's (right).

Future Work. In the future, we would like to leverage other tensor learning scenarios beyond CA –for example, *tensor completion* techniques could fit a compressed tensor using only a fixed amount of known coalition values. Another promising option is to research alternative tensor formats, such as *hierarchical Tucker* (HT), the *multiscale entanglement renormalization ansatz* (MERA), or even more general tensor networks, that could help the method scale to hundreds of players.

References

[1] J. Staudacher, J. Anwander, Using the R package CoopGame for the analysis, solution and visualization of cooperative games with transferable utility, R Vignette (2019).

[2] J. M. Alonso-Meijide, C. Bowles, Generating functions for coalitional power indices: An application to the IMF, Annals of Operations Research 137 (2005) 21–44.

[3] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic, Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Foundations and Trends in Machine Learning 9 (4-5) (2016) 249–429.

[4] R. J. Weber, Probabilistic values for games, Cambridge University Press, 1988, pp. 101–120.

[5] J. von Neumann, O. Morgenstern, Theory of games and economic behavior, Princeton University Press, 1944.

[6] L. S. Shapley, A value for n-person games, in: H. W. Kuhn, A. Tucker (Eds.), Contributions to the Theory of Games (AM-28), Vol. 2, Princeton University Press, 1952, pp. 307–317.

[7] H. Peters, The Shapley Value, Springer Berlin Heidelberg, 2008, pp. 241–258.

[8] A. Saavedra-Nieves, P. Saavedra-Nieves, On systems of quotas from bankruptcy perspective: the sampling estimation of the random arrival rule, European Journal of Operational Research 285 (2) (2020) 655–669.

[9] H. Hamers, R. Lindelauf, B. Husslage, Analysing ISIS Zerkani Network using the Shapley Value, Taylor and Francis, United States, 2019, pp. 463–481.

[10] T. P. Michalak, K. V. Aadithya, P. L. Szczepanski, B. Ravindran, N. R. Jennings, Efficient computation of the Shapley value for game-theoretic network centrality, Journal of Artificial Intelligence Research 46 (1) (2013) 607–650.

[11] J. G. Banzhaf, Weighted voting doesn't work: a mathematical analysis, Rutgers Law Review 19 (1965) 317–343.

[12] Y. Bachrach, E. Markakis, E. Resnick, A. D. Procaccia, J. S. Rosenschein, A. Saberi, Approximating power indices: Theoretical and empirical analysis, Autonomous Agents and Multi-Agent Systems 20 (2) (2010) 105–122.

[13] R. T. Göllner, The Visegrád group – a rising star post-Brexit? Changing distribution of power in the European Council, Open Political Science 1 (1) (2018) 1–6.

[14] A. Puente, Contributions to the representability of simple games and to the calculus of solutions for this class of games, Doctoral thesis, Universitat Politècnica de Catalunya (2000).

[15] F. Carreras, M. Llongueras, M. A. Puente, Partnership formation and binomial semivalues, European journal of operational research 192 (2) (2009) 487–499.

[16] F. Carreras, M. A. Puente, Symmetric coalitional binomial semivalues, Group Decision and Negotiation 21 (2011) 637–662.

[17] Y. Wang, D. Cheng, X. Liu, Matrix expression of Shapley values and its application to distributed resource allocation, Science China Information Sciences 62 (022201) (2019) 1–11.

[18] D. Cheng, H. Qi, Y. Zhao, An introduction to semi-tensor product of matrices and its applications, World Scientific, Singapore, 2012.

[19] J. Castro, D. Gómez, J. Tejada, Polynomial calculation of the Shapley value based on sampling, Computers and Operations Research 36 (5) (2009) 1726–1730.

[20] A. Saavedra-Nieves, I. García-Jurado, M. G. Fiestras-Janeiro, Estimation of the Owen Value Based on Sampling, Springer International Publishing, Cham, 2018, pp. 347–356.

[21] A. Saavedra-Nieves, M. G. Fiestras-Janeiro, Sampling methods to estimate the Banzhaf-Owen value, Annals of Operations Research 301 (2021) 199–223.

[22] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM Review 51 (3) (2009) 455–500.

[23] N. D. Sidiropoulos, L. de Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, C. Faloutsos, Tensor decomposition for signal processing and machine learning, IEEE Transactions on Signal Processing 65 (13) (2017) 3551–3582.

[24] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317.

[25] L. de Lathauwer, B. de Moor, J. Vandewalle, On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors, SIAM Journal of Matrix Analysis and Applications 21 (4) (2000) 1324–1342.

[26] F. L. Hitchcock, The expression of a tensor or a polyadic as a sum of products, Journal of Mathematics and Physics 6 (1927) 164–169.

[27] N. Megiddo, Tensor decomposition of cooperative games, SIAM Journal on Applied Mathematics 29 (3) (1975) 388–405.

[28] G. Owen, The tensor composition of non-negative games, Annals of Mathematics Studies (52) (1964) 307–326.

[29] G. Owen, Multilinear extensions of games, Management Science 18 (5) (1972) P64–P79.

[30] Z. Huang, L. Qi, Formulating an n-person noncooperative game as a tensor complementarity problem, Computational Optimization and Applications 66 (2017) 557–576.

[31] E. Tal, A. A. Gorodetsky, S. Karaman, Continuous tensor train-based dynamic programming for high-dimensional zero-sum differential games, in: Annual American Control Conference, 2018, pp. 6086–6093.

[32] A. Gorodetsky, S. Karaman, Y. Marzouk, High-dimensional stochastic optimal control using continuous tensor decompositions, The International Journal of Robotics Research 37 (2-3) (2018) 340–377.

[33] R. Ballester-Ripoll, E. G. Paredes, R. Pajarola, Tensor algorithms for advanced sensitivity metrics, SIAM/ASA Journal on Uncertainty Quantification 6 (3) (2018) 1172–1197.

[34] I. M. Sobol', Sensitivity estimates for nonlinear mathematical models (in Russian), Mathematical Models 2 (1990) 112–118.

[35] I. V. Oseledets, E. Tyrtyshnikov, TT-cross approximation for multidimensional arrays, Linear Algebra Applications 432 (1) (2010) 70–88.

[36] S. A. Goreinov, E. E. Tyrtyshnikov, The maximal-volume concept in approximation by low-rank matrices, Contemporary Mathematics 280 (2001) 47–51.

[37] D. Savostyanov, Quasioptimality of maximum-volume cross interpolation of tensors, Linear Algebra and its Applications 458 (2014) 217–244.

[38] S. V. Dolgov, B. N. Khoromskij, A. Litvinenko, H. G. Matthies, Computation of the response surface in the tensor train data format, ArXiv e-print 1406.2816.

[39] I. V. Oseledets, Approximation of $2^d \times 2^d$ matrices using tensor decomposition, SIAM Journal on Matrix Analysis and Applications 31 (4) (2010) 2130–2145.

[40] S. C. Littlechild, G. Owen, A simple expression for the Shapley value in a special case, INFORMS Management Science 20 (3) (1973) 370–372.

[41] A. Schotter, Auctioning Böhm-Bawerk's horses, International Journal of Game Theory 3 (1974) 195–215.

[42] R. J. Aumann, M. Maschler, Game theoretic analysis of a bankruptcy problem from the Talmud, Journal of Economic Theory 36 (2) (1985) 195–213.

[43] M. Le Breton, M. Montero, V. Zaporozhets, Voting power in the EU council of ministers and fair decision making in distributive politics, Mathematical Social Sciences 63 (2) (2012) 159–173, around the Cambridge Compromise: Apportionment in Theory and Practice.