SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2019, Vol. 62 022201:1–022201:11
https://doi.org/10.1007/s11432-018-9414-5

Matrix expression of Shapley values and its application to distributed resource allocation

Yuanhua WANG[1*], Daizhan CHENG[2] & Xiyu LIU[1]

[1]*School of Management Science and Engineering, Shandong Normal University, Jinan* 250014, *China;*
[2]*Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing* 100190, *China*

Received 27 January 2018/Accepted 5 March 2018/Published online 25 December 2018

Abstract The symmetric and weighted Shapley values for cooperative n-person games are studied. Using the semi-tensor product of matrices, it is first shown that a characteristic function can be expressed as a pseudo-Boolean function. Then, two simple matrix formulas are obtained for calculating the symmetric and weighted Shapley values. Finally, using these new formulas, a design technique for the agents' payoff functions in distributed resource allocation problems is proposed. It is possible to design payoff functions with the weighted Shapley value by the nonsymmetric weights defined on the players, thus ensuring that the optimal allocation is a pure Nash equilibrium. Practical examples are presented to illustrate the theoretical results.

Keywords semi-tensor product of matrices, Shapley value, matrix formula, distributed resource allocation

Citation Wang Y H, Cheng D Z, Liu X Y. Matrix expression of Shapley values and its application to distributed resource allocation. Sci China Inf Sci, 2019, 62(2): 022201, https://doi.org/10.1007/s11432-018-9414-5

1 Introduction

Game-theoretic control has attracted a great deal of attention in distributed resource allocation problems, such as distributed power control [1] and sensor coverage [2]. In [3], an architectural view was presented to illustrate this approach using potential games as the interface. This architecture allows a variety of payoff function designs at the top layer and learning rule designs at the bottom layer. To design payoff functions, Ref. [4] introduced certain natural approaches to distribution rules, such as equally shared, wonderful life, and Shapley value payoffs.

The Shapley value has long been widely used in cooperative games. It was first introduced by Shapley [5] and was recognized as a sensible method for distributing the profit of cooperation among the players [6]. However, the assumption that all players have equal power may be unrealistic in several applications. For example, in an open market, players may have different bargaining abilities. To resolve this, the weighted Shapley value, a generalization of the symmetric Shapley value, was proposed by Shapley [7]. In economic systems, the weighted Shapley value arises from nonsymmetric divisions of the surplus. There is an extensive literature on the characterizations of the weighted Shapley value [8–10].

Despite the widespread use of the Shapley value, its computation is highly complicated [4]. Particularly, in large scale cases, computational complexity is a serious problem. In this study, two simple matrix formulas are obtained for calculating the symmetric and weighted Shapley values, thereby reducing computational complexity. Then, their application to distributed resource allocation problems is considered, and certain desirable properties are explored. The basic mathematical technique for this approach is the

* Corresponding author (email: wyh_1005@163.com)

semi-tensor product of matrices, which is a generalization of the usual matrix product [11]. It has been successfully applied to the analysis of logical networks [12–18], graph theory [19, 20], and colored petri nets [21]. Recently, it has also been used in cooperative games [22], noncooperative games [23, 24], and evolutionary games [25–27].

The remainder of this paper is organized as follows. In Section 2, a brief review for the semi-tensor product of matrices and cooperative games is presented. In Section 3, two matrix formulas are obtained for calculating the symmetric and weighted Shapley values. In Section 4, the application of Shapley values to distributed resource allocation problems is discussed. Section 5 concludes the paper.

2 Preliminaries

Some notations are first given.

- $\mathcal{M}_{m\times n}$: the set of $m\times n$ real matrices.
- $\mathrm{Col}(M)$: the set of columns of M; $\mathrm{Col}_i(M)$ is the i-th column of M.
- $\mathcal{D}:=\{0,1\}$.
- δ_k^i: the i-th column of the identity matrix I_k.
- $\Delta_k:=\left\{\delta_k^i|i=1,\ldots,k\right\}$.
- $\mathbf{1}_k:=[\underbrace{1,1,\ldots,1}_{k}]^{\mathrm{T}}$.

2.1 Semi-tensor product of matrices

The basic technique used in this study is the semi-tensor product of matrices [11].

Definition 1. Let $A\in\mathcal{M}_{m\times n}$, $B\in\mathcal{M}_{p\times q}$. Moreover, $t=\mathrm{lcm}\{n,p\}$ denotes the least common multiple of n and p. Then, the semi-tensor product (STP) of A and B is defined by

$$A\ltimes B:=\left(A\otimes I_{t/n}\right)\left(B\otimes I_{t/p}\right)\in\mathcal{M}_{mt/n\times qt/p},\tag{1}$$

where $\otimes$ is the Kronecker product.

The semi-tensor product is a generalization of the usual matrix product, and the symbol "$\ltimes$" may be omitted without confusion. The semi-tensor product has pseudo-commutative properties, and the key to these properties is the swap matrix.

Definition 2. The matrix $W_{[m,n]}\in\mathcal{M}_{mn\times mn}$, defined by

$$W_{[m,n]}=\delta_{mn}[1,m+1,\ldots,(n-1)m+1;2,m+2,\ldots,(n-1)m+2;\ \ldots\ ;m,2m,\ldots,nm],$$

is called the (m,n)-th dimensional swap matrix.

Proposition 1. Let $X\in\mathbb{R}^m$ and $Y\in\mathbb{R}^n$ be two columns. Then

$$W_{[m,n]}\ltimes X\ltimes Y=Y\ltimes X.$$

If $x\in\mathcal{D}$, then its vector form is

$$x\sim\begin{bmatrix}x\\1-x\end{bmatrix}\in\Delta_2.$$

Using the vector form, a pseudo-logical function can be expressed algebraically.

Theorem 1. Let $f:\mathcal{D}^n\to\mathbb{R}$ be a pseudo-logical function. Then, there exists a unique vector $M_f\in\mathbb{R}^{1\times 2^n}$, such that

$$f(x_1,x_2,\ldots,x_n)=M_f\ltimes_{i=1}^n x_i,$$

where $x_i\in\Delta_2$, $i=1,\ldots,n$ and M_f is called the structure vector of f.

2.2 Cooperative games

Definition 3 ([28]). Let (N, v) be an n-person cooperative game, where $N = \{1, 2, \ldots, n\}$ with $v(\emptyset) = 0$ is the set of players. All subsets of N are called coalitions, and N is called the grand coalition. $v : 2^N \to \mathbb{R}$ is called the characteristic function.

For each coalition $R \subseteq N$, to express $v(R)$ using logical variables, let

$$x_i^R = \begin{cases} 1, & i \in R, \\ 0, & i \notin R. \end{cases}$$

Then, each characteristic function $v(R)$ can be expressed as a pseudo-logical function $v : \mathcal{D}^n \to \mathbb{R}$, that is,

$$v(R) = v\left[x_1^R, x_2^R, \ldots, x_n^R\right], \quad R \in 2^N. \tag{2}$$

Using Theorem 1, for each $v(R)$, its structure vector $C_v \in \mathbb{R}^{2^n}$ can be determined, so that (2) may be converted into its algebraic form as follows:

$$v(R) = C_v x^R, \tag{3}$$

where $x^R = \ltimes_{i=1}^n x_i^R \in \Delta_{2^n}$, $C_v := [v[1, 1, \ldots, 1, 1], v[1, 1, \ldots, 1, 0], \ldots, v[0, 0, \ldots, 0, 0]]$. As $v(\emptyset) = 0$, the last element of C_v is 0, that is, $v[0, 0, \ldots, 0, 0] = 0$.

In general, a cooperative game (N, v) is identified with its characteristic function v, and the set of all games is denoted by Γ.

Dividing $v(R)$ among the members of R is an important problem in a cooperative game, and it can be approached using cores, stable sets, or Shapley values.

3 Matrix expression of Shapley values

In this section, two matrix formulas for the symmetric and weighted Shapley values are obtained.

3.1 Matrix formula of symmetric Shapley value

The symmetric Shapley value is defined as follows.

Definition 4 ([5]). Let $v \in \Gamma$ be a cooperative game. The symmetric Shapley value $\varphi : \Gamma \to \mathbb{R}^n$ is an imputation defined as

$$\begin{aligned} \varphi_i(v) &= \frac{1}{n!} \sum_{\sigma \in \pi(N)} \left[v\left(S_\sigma^i \cup \{i\}\right) - v\left(S_\sigma^i\right)\right] \\ &= \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n-1-|S|)!}{n!} \left[v\left(S \cup \{i\}\right) - v\left(S\right)\right], \quad i = 1, 2, \ldots, n, \end{aligned} \tag{4}$$

where $\pi(N)$ is the set of all permutations for $N = \{1, 2, \ldots, n\}$. S_σ^i denotes the set of players preceding i in the permutation σ, that is, $S_\sigma^i = \{j | \sigma^{-1} j < \sigma^{-1} i\}$.

Using (3), we have

$$\begin{aligned} v(S \cup \{i\}) - v(S) &= C_v \left(x_1^S \cdots x_{i-1}^S \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_{i+1}^S \cdots x_n^S - x_1^S \cdots x_{i-1}^S \begin{bmatrix} 0 \\ 1 \end{bmatrix} x_{i+1}^S \cdots x_n^S \right) \\ &= C_v \left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_1^S \cdots x_{i-1}^S x_{i+1}^S \cdots x_n^S - W_{[2,2^{i-1}]} \begin{bmatrix} 0 \\ 1 \end{bmatrix} x_1^S \cdots x_{i-1}^S x_{i+1}^S \cdots x_n^S \right) \\ &= C_v \left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} x_1^S \cdots x_{i-1}^S x_{i+1}^S \cdots x_n^S \right), \end{aligned}$$

where $\ltimes_{j\neq i} x_j^S \in \Delta_{2^{n-1}}$, and

$$x_j^S = \begin{cases} \delta_2^1, & j \in S, \\ \delta_2^2, & j \notin S. \end{cases}$$

Then,

$$x_1^S \cdots x_{i-1}^S x_{i+1}^S \cdots x_n^S = \delta_{2^{n-1}}^j, \quad j = 1, 2, \ldots, 2^{n-1}.$$

A vector ℓ_j is now recursively constructed as follows:

$$\begin{cases} \ell_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\ \ell_{j+1} = \begin{bmatrix} \ell_j + \mathbf{1}_{2^j} \\ \ell_j \end{bmatrix} \in \mathbb{R}^{2^{j+1}}, \quad j = 1, 2, 3, \ldots. \end{cases}$$

Let ℓ_k^i denote the i-th component of ℓ_k, and let

$$\ell_k^i = \left|\delta_{2^k}^i\right| = |S|, \quad i = 1, 2, \ldots, 2^k,$$

where $x^S = \ltimes_{i=1}^k x_i^S = \delta_{2^k}^i$. Then, a column vector $\eta_k = [\eta_k^1, \eta_k^2, \ldots, \eta_k^{2^k}]^{\mathrm{T}}$ is constructed, where

$$\eta_k^i = \left(\ell_k^i\right)! \left(k - \ell_k^i\right)!, \quad i = 1, \ldots, 2^k.$$

By the above analysis, Eq. (4) can be expressed as

$$\varphi_i(v) = \frac{1}{n!} C_v \sum_{j=1}^{2^{n-1}} \eta_{n-1}^j W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \delta_{2^{n-1}}^j, \quad i = 1, \ldots, n. \tag{5}$$

According to Definition 2, we have

$$W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = \operatorname{diag}\left(\underbrace{\begin{bmatrix} 1 \\ -1 \end{bmatrix}, \ldots, \begin{bmatrix} 1 \\ -1 \end{bmatrix}}_{2^{i-1}} \right) \in \mathcal{M}_{2^i \times 2^{i-1}}.$$

It is clear that

$$\left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \ltimes \delta_{2^{n-1}}^j = \left(\left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \otimes I_{2^{n-i}} \right) \delta_{2^{n-1}}^j = \operatorname{Col}_j(T_i),$$

where

$$T_i = \left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) \otimes I_{2^{n-i}} = \operatorname{diag}\left(\underbrace{\begin{bmatrix} I_{2^{n-i}} \\ -I_{2^{n-i}} \end{bmatrix}, \ldots, \begin{bmatrix} I_{2^{n-i}} \\ -I_{2^{n-i}} \end{bmatrix}}_{2^{i-1}} \right) \in \mathcal{M}_{2^n \times 2^{n-1}}. \tag{6}$$

A new vector $\varepsilon = \eta_{n-1}$ is now defined, and ε is divided into m equal blocks as follows:

$$\varepsilon = \varepsilon_1^1 = \left[\varepsilon_2^1, \varepsilon_2^2\right]^{\mathrm{T}} = \cdots = \left[\varepsilon_m^1, \varepsilon_m^2, \ldots, \varepsilon_m^m\right]^{\mathrm{T}}, \quad m = 1, 2, 2^2, \ldots, 2^{n-1}.$$

Then, it is easy to verify that

$$\sum_{j=1}^{2^{n-1}} \eta_{n-1}^j W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \delta_{2^{n-1}}^j = \left[\varepsilon_{2^{i-1}}^1, -\varepsilon_{2^{i-1}}^1, \varepsilon_{2^{i-1}}^2, -\varepsilon_{2^{i-1}}^2, \ldots, \varepsilon_{2^{i-1}}^{2^{i-1}}, -\varepsilon_{2^{i-1}}^{2^{i-1}}\right]^{\mathrm{T}}.$$

Hence, a new matrix formula is obtained for calculating the symmetric Shapley value using the semi-tensor product as follows.

Theorem 2 ([22]). The Shapley value $\varphi(v) = [\varphi_1(v), \ldots, \varphi_n(v)]$ of a game $v \in \Gamma$ can be calculated by

$$\varphi(v) = C_v \Phi_n, \tag{7}$$

where

$$\Phi_n = \frac{1}{n!} \begin{bmatrix} & & \varepsilon_{2^{n-1}}^1 \\ & & -\varepsilon_{2^{n-1}}^1 \\ & \varepsilon_2^1 & \varepsilon_{2^{n-1}}^2 \\ \varepsilon_1^1 & -\varepsilon_2^1 & \cdots & -\varepsilon_{2^{n-1}}^2 \\ -\varepsilon_1^1 & \varepsilon_2^2 & & \vdots \\ & -\varepsilon_2^2 & & \varepsilon_{2^{n-1}}^{2^{n-1}} \\ & & & -\varepsilon_{2^{n-1}}^{2^{n-1}} \end{bmatrix} \in \mathcal{M}_{2^n \times n}. \tag{8}$$

3.2 Matrix formula of weighted Shapley value

The weighted Shapley value is a natural generalization of the symmetric Shapley value and can be used for fairly dividing up $v(N)$ by assigning to each player a positive weight. Let any permutation $\sigma \in \pi(N)$ be denoted by $\sigma = [i_1, i_2, \ldots, i_n]$, which implies that $\sigma(k) = i_k$, $k = 1, \ldots, n$. The weighted Shapley value can then be defined as follows.

Definition 5 ([9]). Let $v \in \Gamma$ be a cooperative game. Given a vector of positive weights $\lambda = [\lambda_1, \ldots, \lambda_n] \in \mathbb{R}_+^n$ such that $\sum_{i=1}^n \lambda_i = 1$, the weighted Shapley value $\varphi^\lambda : \Gamma \to \mathbb{R}^n$ is defined by

$$\varphi_i^\lambda(v) = \sum_{\sigma \in \pi(N)} P_\lambda(\sigma) \left[v\left(S_\sigma^i \cup \{i\}\right) - v\left(S_\sigma^i\right) \right], \quad i = 1, 2, \ldots, n, \tag{9}$$

where $P_\lambda(\sigma) = \prod_{k=1}^n \frac{\lambda_{i_k}}{\sum_{t=1}^k \lambda_{i_t}}$ is a probability distribution over $\sigma \in \pi(N)$, and obviously, $\sum_{\sigma \in \pi(N)} P_\lambda(\sigma) = 1$. If $\lambda_i = \frac{1}{n}$ for all i, then $\varphi_i^\lambda(v) = \varphi_i(v)$.

For $N = \{1, 2, \ldots, n\}$, all its permutations are considered. In combinatorics, the lexicographic order method is widely used for systematically generating permutations [29].

In this approach, there is a one-to-one mapping between permutations and ordered sequences. For example, given $N = \{1, 2, 3, 4\}$, the first sequence is $(1, 2, 3, 4)$, and the last is $(4, 3, 2, 1)$. Algorithm 1 provides a useful method for lexicographically ordering sequences [30].

Algorithm 1

Let $\sigma = (i_1, i_2, \ldots, i_n)$ be a permutation for N.

1: Start at the right end of σ. Find the largest index j such that $i_j < i_{j+1}$. If no such index exists, then σ is the last permutation.

2: From the right-hand side of i_j, find the largest index k greater than j such that $i_j < i_k$.

3: Swap the value of i_j with that of i_k;

4: Reverse the sequence $i_{j+1}, \ldots, i_{k-1}, i_k, i_{k+1}, \ldots, i_n$, and then the next permutation after σ is

$$\sigma' = (i_1, i_2, \ldots, i_{j-1}, i_j, i_n, \ldots, i_{k+1}, i_k, i_{k-1}, \ldots, i_{j+1}).$$

Using Algorithm 1, all permutations of N can be obtained in an order $\prec$. More precisely, $\sigma = (i_1, i_2, \ldots, i_n) \prec \mu = (j_1, j_2, \ldots, j_n)$ if and only if there exists a $1 \leqslant k \leqslant n$ such that

$$\begin{cases} i_s = j_s, \\ i_k < j_k, \end{cases} \quad s < k.$$

A probability vector Ξ is now defined by

$$\Xi = [P_\lambda(\pi_1), P_\lambda(\pi_2), \ldots, P_\lambda(\pi_{n!})], \tag{10}$$

where π_p $(p = 1, 2, \ldots, n!)$ is the p-th permutation.

In vector form, the weighted Shapley value (9) can be rewritten as follows:

$$\begin{aligned}
\varphi_i^{\lambda}(v) &= \sum_{p=1}^{n!} P_{\lambda}(\pi_p)\left(v\left(S_{\pi_p}^i \cup \{i\}\right) - v\left(S_{\pi_p}^i\right)\right) \\
&= \sum_{p=1}^{n!} P_{\lambda}(\pi_p) C_v \left(x_1^{S_{\pi_p}^i} \cdots x_{i-1}^{S_{\pi_p}^i} \begin{bmatrix} 1 \\ 0 \end{bmatrix} x_{i+1}^{S_{\pi_p}^i} \cdots x_n^{S_{\pi_p}^i} - x_1^{S_{\pi_p}^i} \cdots x_{i-1}^{S_{\pi_p}^i} \begin{bmatrix} 0 \\ 1 \end{bmatrix} x_{i+1}^{S_{\pi_p}^i} \cdots x_n^{S_{\pi_p}^i} \right) \\
&= \sum_{p=1}^{n!} P_{\lambda}(\pi_p) C_v \left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} x_1^{S_{\pi_p}^i} \cdots x_{i-1}^{S_{\pi_p}^i} x_{i+1}^{S_{\pi_p}^i} \cdots x_n^{S_{\pi_p}^i} \right) \\
&= C_v \sum_{p=1}^{n!} P_{\lambda}(\pi_p) \left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} x_1^{S_{\pi_p}^i} \cdots x_{i-1}^{S_{\pi_p}^i} x_{i+1}^{S_{\pi_p}^i} \cdots x_n^{S_{\pi_p}^i} \right),
\end{aligned}$$

where $S_{\pi_p}^i$ is the set of players preceding i in permutation π_p, that is, $S_{\pi_p}^i = \{j | \pi_p^{-1} j < \pi_p^{-1} i\}$. Let

$$x_1^{S_{\pi_p}^i} \ltimes \cdots \ltimes x_{i-1}^{S_{\pi_p}^i} \ltimes x_{i+1}^{S_{\pi_p}^i} \ltimes \cdots \ltimes x_n^{S_{\pi_p}^i} = \delta_{2^{n-1}}^{j_{p,i}}, \quad i = 1, 2, \ldots, n, \quad p = 1, 2, \ldots, n!, \tag{11}$$

where

$$x_j^{S_{\pi_p}^i} = \begin{cases} \delta_2^1, & j \in S_{\pi_p}^i, \\ \delta_2^2, & j \notin S_{\pi_p}^i. \end{cases}$$

As in the proof of Theorem 2, we have

$$\begin{aligned}
\varphi_i^{\lambda}(v) &= C_v \sum_{p=1}^{n!} P_{\lambda}(\pi_p) \left(W_{[2,2^{i-1}]} \begin{bmatrix} 1 \\ -1 \end{bmatrix} \delta_{2^{n-1}}^{j_{p,i}} \right) \\
&= C_v \sum_{p=1}^{n!} P_{\lambda}(\pi_p) \operatorname{Col}_{j_{p,i}}(T_i).
\end{aligned}$$

From the above analysis, Theorem 3 can be obtained.

Theorem 3. Let $v \in \Gamma$ be a cooperative game. Then, its weighted Shapley value $\varphi^{\lambda}(v) = [\varphi_1^{\lambda}(v), \ldots, \varphi_n^{\lambda}(v)]$ can be calculated by

$$\varphi^{\lambda}(v) = C_v(\Xi\Psi), \tag{12}$$

where

$$\Psi = \begin{bmatrix} \operatorname{Col}_{j_{1,1}}(T_1) & \operatorname{Col}_{j_{1,2}}(T_2) & \cdots & \operatorname{Col}_{j_{1,n}}(T_n) \\ \operatorname{Col}_{j_{2,1}}(T_1) & \operatorname{Col}_{j_{2,2}}(T_2) & \cdots & \operatorname{Col}_{j_{2,n}}(T_n) \\ \vdots & \vdots & & \vdots \\ \operatorname{Col}_{j_{n!,1}}(T_1) & \operatorname{Col}_{j_{n!,2}}(T_2) & \cdots & \operatorname{Col}_{j_{n!,n}}(T_n) \end{bmatrix} \in \mathcal{M}_{(n!2^n)\times n}. \tag{13}$$

Remark 1. Comparing (7) and (12), it can be seen that the calculation of the weighted Shapley value is more complicated than that of the symmetric Shapley value. This is primarily due to the computation complexity of the matrix Ψ. Hence, the proposed formula (12) in Theorem 3 can only handle small size cases. However, there is room for further improvement.

Example 1 (The horse trading problem)**.** Player 1 is selling a horse. Players 2 and 3 intend to buy a horse for at most 90 dollars and 100 dollars, respectively. It is assumed that these three players have different bargaining abilities, which can be expressed as weights. It is now assumed that $\lambda = [0.4,\ 0.1,\ 0.5]$.

This problem can be modeled as a cooperative game, where $N=\{1,2,3\}$, $v(\{1\})=v(\{2\})=v(\{3\})=v(\{2,3\})=0$, $v(\{1,2\})=90$, $v(\{1,3\})=100$, and $v(\{1,2,3\})=100$. The characteristic function can be converted into its algebraic form as

$$v(R)=C_v x_1^R x_2^R x_3^R=[100,90,100,0,0,0,0,0]\,x^R,$$

where $x^R=\ltimes_{i=1}^3 x_i^R\in\Delta_{2^3}$.

Using Algorithm 1, all permutations of $N=\{1,2,3\}$ are listed as follows:

$$\pi_1=(1,2,3),\ \ \pi_2=(1,3,2),\ \ \pi_3=(2,1,3),\ \ \pi_4=(2,3,1),\ \ \pi_5=(3,1,2),\ \ \pi_6=(3,2,1).$$

According to Definition 5,

$$\Xi=[P_\lambda(\pi_1),\ P_\lambda(\pi_2),\ P_\lambda(\pi_3),\ P_\lambda(\pi_4),\ P_\lambda(\pi_5),\ P_\lambda(\pi_6)]=[0.1,\ 0.056,\ 0.4,\ 0.333,\ 0.044,\ 0.067].$$

Obviously, $\sum_{p=1}^6 P_\lambda(\pi_p)=1$.

Using (11), it is easy to determine the index $j_{p,i}\in\{1,2,3,4\}$, where $i=1,2,3$, and $p=1,2,\ldots,3!$. Then, according to (13), the following matrix is constructed

$$\Psi=\begin{bmatrix}\mathrm{Col}_4(T_1)\ \mathrm{Col}_2(T_2)\ \mathrm{Col}_1(T_3)\\ \mathrm{Col}_4(T_1)\ \mathrm{Col}_1(T_2)\ \mathrm{Col}_2(T_3)\\ \mathrm{Col}_2(T_1)\ \mathrm{Col}_4(T_2)\ \mathrm{Col}_1(T_3)\\ \mathrm{Col}_1(T_1)\ \mathrm{Col}_4(T_2)\ \mathrm{Col}_3(T_3)\\ \mathrm{Col}_3(T_1)\ \mathrm{Col}_1(T_2)\ \mathrm{Col}_4(T_3)\\ \mathrm{Col}_1(T_1)\ \mathrm{Col}_3(T_2)\ \mathrm{Col}_4(T_3)\end{bmatrix},$$

where T_i, $i=1,2,3$, is calculated by (6). Using the matrix formula (12), the weighted Shapley value is

$$\varphi^\lambda(v)=C_v(\Xi\Psi)=[80.4,\ 9,\ 10.6].$$

It is obvious that $\sum_{i=1}^3\varphi_i^\lambda(v)=100=v(\{1,2,3\})$, which conforms with the efficiency axiom. Moreover, compared with the symmetric Shapley value $\varphi(v)=[65,\ 15,\ 20]$ for this problem [22], the weighted Shapley value $\varphi^\lambda(v)$ appears to be more sensible and realistic in this market trading, where players 1 and 3 tend to cooperate. This advantage makes the weighted Shapley value a powerful tool in practical applications.

4 Application to distributed resource allocation

In this section, the application of Shapley value to the distributed resource allocation problem is investigated.

Definition 6 ([3])**.** A resource allocation game is defined by the tuple $G=(N,R,\mathcal{A},C,W)$.

(i) $N=\{1,2,\ldots,n\}$ is the set of distributed agents. The set of resources that will be shared by the agents is denoted by $R=\{r_1,r_2,\ldots,r_m\}$.

(ii) $\mathcal{A}=\prod_{i=1}^n\mathcal{A}_i$ is the set of possible allocations, where $\mathcal{A}_i\subseteq 2^R$ is the action set for agent i, that is, each agent $i\in N$ can select multiple resources in R. The allocation of all agents but the i-th is denoted by $a_{-i}\in\mathcal{A}_{-i}:=\prod_{j\neq i}\mathcal{A}_j$.

(iii) $C=[c_1,\ldots,c_n]\in\mathbb{R}^n$ with $c_i:\mathcal{A}\to\mathbb{R}$ is the payoff function for agent i.

(iv) $W(a):\mathcal{A}\to\mathbb{R}$ is a global welfare function that should be optimized. It is assumed that $W(a)=\sum_{r\in R}W_r(\{a\}_r)$, where $\{a\}_r=\{i\in N: r\in a_i\}$ is the set of agents that are allocated to resource r in $a\in\mathcal{A}$, and $W_r:2^N\to\mathbb{R}$ is the local welfare function for resource r.

The aim is to design a payoff function for each player, so that a resource allocation game becomes a potential game. Then, as the players maximize their payoff functions, the global welfare function can be optimized.

The design of a payoff function $c_i(a_i, a_{-i})$ with the following form is first considered:

$$c_i(a_i, a_{-i}) = \sum_{r \in a_i} \mathrm{Sh}_i(\{a\}_r, W_r), \tag{14}$$

where $\mathrm{Sh}_i(\{a\}_r, W_r)$ is the Shapley value of player i at resource r, that is,

$$\mathrm{Sh}_i(\{a\}_r, W_r) = \sum_{S \subseteq \{a\}_r \setminus \{i\}} \frac{|S|!(|\{a\}_r| - |S| - 1)!}{|\{a\}_r|!} \left[W_r(S \cup \{i\}) - W_r(S)\right]. \tag{15}$$

It is assumed that $\{a\}_r = \{k_1, k_2, \ldots, k_l\}$, $l = |\{a\}_r|$. According to (3), the structure vector $C_{W_r} \in \mathbb{R}^{2^l}$ of the local welfare function $W_r(S)$ can be determined, so that

$$W_r(S) = C_{W_r} \ltimes_{i=1}^{l} a_{k_i}^S,$$

where

$$C_{W_r} = [W_r(\{k_1, k_2, \ldots, k_l\}), W_r(\{k_1, k_2, \ldots, k_{l-1}\}), \ldots, W_r(\{k_l\}), W_r(\emptyset)],$$

and $a_{k_i}^S \in \Delta_2$, $i = 1, 2, \ldots, l$.

Using Theorem 2, we have Proposition 2.

Proposition 2. The payoff function (14) of a resource allocation game $G = (N, R, \mathcal{A}, C, W)$ can be calculated by

$$c_i(a_i, a_{-i}) = \sum_{r \in a_i} \mathrm{Sh}_i(\{a\}_r, W_r) = \sum_{r \in a_i} C_{W_r} \mathrm{Col}_i(\Phi_l^r), \tag{16}$$

where $\Phi_l^r \in \mathcal{M}_{2^l \times l}$ can be constructed using (8).

The payoff function $c_i(a_i, a_{-i})$ with the following weighted Shapley value form is now considered:

$$c_i(a_i, a_{-i}) = \sum_{r \in a_i} \mathrm{Sh}_i^{\lambda}(\{a\}_r, W_r). \tag{17}$$

It is assumed that any permutation on $\{a\}_r$ is denoted by $\alpha^r = (i_{k_1}, i_{k_2}, \ldots, i_{k_l}) \in \pi(\{a\}_r)$. Then, according to Definition 5, we have

$$\mathrm{Sh}_i^{\lambda}(\{a\}_r, W_r) = \sum_{\alpha^r \in \pi(\{a\}_r)} P_{\lambda}(\alpha^r) \left[W_r(S_{\alpha^r}^i \cup \{i\}) - W_r(S_{\alpha^r}^i)\right], \tag{18}$$

where $\lambda = [\lambda_{k_1}, \ldots, \lambda_{k_l}] \in \mathbb{R}_+^l$, so that $\sum_{i=1}^{l} \lambda_{k_i} = 1$. $S_{\alpha^r}^i$ is the set of players preceding i in the permutation α^r.

Let

$$\Xi^r = [P_{\lambda}(\pi_1^r), P_{\lambda}(\pi_2^r), \ldots, P_{\lambda}(\pi_{l!}^r)],$$

where π_p^r $(p = 1, 2, \ldots, l!)$ is the p-th permutation of $\pi(\{a\}_r)$.

Using the matrix formula (12) in Theorem 3, we have Proposition 3.

Proposition 3. The payoff function (17) of $G = (N, R, \mathcal{A}, C, W)$ can be calculated by

$$c_i(a_i, a_{-i}) = \sum_{r \in a_i} \mathrm{Sh}_i^{\lambda}(\{a\}_r, W_r) = \sum_{r \in a_i} C_{W_r}[\Xi^r \mathrm{Col}_i(\Psi^r)], \tag{19}$$

where $\Psi^r \in \mathcal{M}_{(l!2^l) \times l}$ can be constructed using (13).

In [4], it was proved that both the symmetric and weighted Shapley values can be applied to a distributed resource allocation game, which can lead to a potential game, thus ensuring the existence of pure Nash equilibria.

Remark 2. By introducing exogenous weights, the weighted Shapley value can ensure that the Nash equilibria are efficient with respect to the global objective. That is, the Nash equilibria can be refined to some extent by choosing proper weights, ensuring that the refined Nash equilibrium is the optimal allocation.

Table 1 Payoff function using symmetric Shapley value

	r_2	r_3
r_1	1, 1	1, 0.5
r_2	1.5, 0.5	2, 0.5

Example 2. A classical vehicle target assignment problem is considered, where there are: (i) vehicles $N = \{1, 2\}$, (ii) invariant detection probabilities $p_1 = 1$ and $p_2 = 0.5$, (iii) targets (or resources) $R = \{r_1, r_2, r_3\}$, (iv) values $w_{r_1} = 1$, $w_{r_2} = 2$, and $w_{r_3} = 1$, and (v) action sets $\mathcal{A}_1 = \{r_1, r_2\}$ and $\mathcal{A}_2 = \{r_2, r_3\}$. The aim here is to develop a method for designing a payoff function ensuring the existence of pure Nash equilibria and the convergence of the global behavior to a^* that maximizes the global objective

$$W(a) = \sum_{r\in R;\{a\}_r\neq\emptyset} w_r \left(1 - \prod_{i\in\{a\}_r} (1 - p_i)\right). \tag{20}$$

The symmetric Shapley value form (14) is first used. By (16),

(1) When $a_1 = r_1$, $a_2 = r_2$, we have

$$\begin{aligned} C_{W_{r_1}} &= [W_{r_1}(\{1\}),\ W_{r_1}(\{\emptyset\})] = [w_{r_1}(1 - (1 - p_1)),\ 0] = [1,\ 0], \\ C_{W_{r_2}} &= [W_{r_2}(\{2\}),\ W_{r_2}(\{\emptyset\})] = [w_{r_2}(1 - (1 - p_2)),\ 0] = [1,\ 0]. \end{aligned}$$

Then,

$$\begin{aligned} c_1(a_1, a_2) &= \mathrm{Sh}_1(\{a\}_{r_1}, W_{r_1}) = C_{W_{r_1}} \mathrm{Col}_1(\Phi_1^{r_1}) = 1, \\ c_2(a_2, a_1) &= \mathrm{Sh}_2(\{a\}_{r_2}, W_{r_2}) = C_{W_{r_2}} \mathrm{Col}_1(\Phi_1^{r_2}) = 1, \end{aligned}$$

where $\Phi_1^{r_1} = \Phi_1^{r_2} = [1,\ -1]^{\mathrm{T}}$.

(2) When $a_1 = r_2$, $a_2 = r_2$, it follows that $\{a\}_{r_2} = \{1, 2\}$; thus,

$$\begin{aligned} C_{W_{r_2}} &= [W_{r_2}(\{1, 2\}),\ W_{r_2}(\{1\}),\ W_{r_2}(\{2\}),\ W_{r_2}(\{\emptyset\})] \\ &= [w_{r_2}(1 - (1 - p_1)(1 - p_2)),\ w_{r_2}(1 - (1 - p_2)),\ w_{r_2}(1 - (1 - p_1)),\ 0] \\ &= [2,\ 2,\ 1,\ 0]. \end{aligned}$$

Then,

$$\begin{aligned} c_1(a_1, a_2) &= \mathrm{Sh}_1(\{a\}_{r_2}, W_{r_2}) = C_{W_{r_2}} \mathrm{Col}_1(\Phi_2^{r_2}) = 1.5, \\ c_2(a_2, a_1) &= \mathrm{Sh}_2(\{a\}_{r_2}, W_{r_2}) = C_{W_{r_2}} \mathrm{Col}_2(\Phi_2^{r_2}) = 0.5, \end{aligned}$$

where

$$\Phi_2^{r_2} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & -1 \end{bmatrix}.$$

The payoff functions for $(a_1, a_2) = (r_1, r_3)$ and $(a_1, a_2) = (r_2, r_3)$ are similarly calculated, as shown in Table 1.

It is clear that this game has two pure Nash equilibria (r_2, r_2) and (r_2, r_3). Using the method in [23], it is easy to verify that it is a potential game, and its potential function is

$$P(a) = [0,\ -0.5,\ 0.5,\ 0.5] \ltimes_{i=1}^{2} a_i + c_0, \quad \forall c_0 \in \mathbb{R}.$$

The value of $W(a)$ can be calculated using (20). This is listed in Table 2, where it can easily be seen that the optimal allocation (r_2, r_3) is that of pure Nash equilibria; thus, the convergence of the global behavior to the optimal allocation cannot be ensured.

Table 2 Global objective $W(a)$

	r_2	r_3
r_1	2	1.5
r_2	2	2.5

Table 3 Payoff function using the weighted Shapley value

	r_2	r_3
r_1	1, 1	1, 0.5
r_2	1.8, 0.2	2, 0.5

The weighted Shapley value form (18) is now considered. It is assumed that $(\lambda_1, \lambda_2) = (0.2, 0.8)$. Then,

$$\Xi^{r_2} = [P_\lambda(\pi_1^{r_2}),\ P_\lambda(\pi_2^{r_2})] = \left[\frac{\lambda_2}{\lambda_1+\lambda_2},\ \frac{\lambda_1}{\lambda_1+\lambda_2}\right] = [0.8,\ 0.2].$$

It should be noted that the weights affect the payoff function only for $(a_1, a_2) = (r_2, r_2)$. Using (19), we have

$$c_1(a_1, a_2) = \mathrm{Sh}_1^\lambda(\{a\}_{r_2}, W_{r_2}) = C_{W_{r_2}}\left[\Xi^{r_2}\,\mathrm{Col}_1(\Psi^{r_2})\right] = 1.8,$$
$$c_2(a_2, a_1) = \mathrm{Sh}_2^\lambda(\{a\}_{r_2}, W_{r_2}) = C_{W_{r_2}}\left[\Xi^{r_2}\,\mathrm{Col}_2(\Psi^{r_2})\right] = 0.2,$$

where

$$\Psi^{r_2} = \begin{bmatrix}\mathrm{Col}_2(T_1^{r_2})\ \mathrm{Col}_1(T_2^{r_2})\\ \mathrm{Col}_1(T_1^{r_2})\ \mathrm{Col}_2(T_2^{r_2})\end{bmatrix} \in \mathcal{M}_{8\times 2},$$

and $T_i^{r_2} \in \mathcal{M}_{4\times 2}$, $(i = 1, 2)$ is constructed by (6). A new payoff matrix is thus obtained, as shown in Table 3. Obviously, this game has only one pure Nash equilibrium $a^* = (r_2, r_3)$, which is the optimal allocation. Moreover, it can be proved that it is a weighted potential game, and its potential function is

$$P^w(a) = [-3.75,\ -4.375,\ 0.25,\ 0.625] \ltimes_{i=1}^{2} a_i + c_0, \quad \forall c_0 \in \mathbb{R}.$$

It was shown that the weighted Shapley value design can lead to a weighted potential game and ensure that the optimal allocation is the refined pure Nash equilibrium a^*, that is, $a^* = (r_2, r_3) = \mathrm{argmax}_{a\in\mathcal{A}} W(a)$.

Motivated by this example, a simple noncooperative mechanism can be constructed whose outcome always coincides with the Shapley value for cooperative games in economic systems, thus determining how the surplus generated by cooperation is to be shared in a transferable utility environment.

5 Conclusion

A new approach to the symmetric and weighted Shapley values for cooperative games was proposed. Matrix expressions for symmetric and weighted Shapley values were first obtained using the semi-tensor product of matrices, and certain desirable characterizations of them were presented. Finally, the new matrix formulas were extended to study the problem of distributed resource allocation. It was shown that these new matrix expressions not only reduce computational complexity but also provide a convenient design technique for practical applications.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61773371).

References

1 Mei S W, Wang Y Y, Liu F, et al. Game approaches for hybrid power system planning. IEEE Trans Sustain Energ, 2012, 3: 506–517

2 Zhu M H, Martínez S. Distributed coverage games for energy-aware mobile sensor networks. SIAM J Control Optim, 2013, 51: 1–27

3 Gopalakrishnan R, Marden J R, Wierman A. An architectural view of game theoretic control. SIGMETRICS Perform Eval Rev, 2011, 38: 31–36
4 Marden J R, Wierman A. Distributed welfare games. Oper Res, 2013, 61: 155–168
5 Shapley L S. A value for n-person games. Contrib Theory Games, 1953, 2: 307–317
6 Pérez-Castrillo D, Wettstein D. Bidding for the surplus: a non-cooperative approach to the Shapley value. J Econ Theory, 2001, 100: 274–294
7 Shapley L S. Additive and non-additioe set functions. Dissertation for Ph.D. Degree. Princeton: Princeton University, 1953
8 Kalai E, Samet D. On weighted Shapley values. Int J Game Theory, 1987, 16: 205–222
9 Chun Y. On the symmetric and weighted shapley values. Int J Game Theory, 1991, 20: 183–190
10 Nowak A S, Radzik T. On axiomatizations of the weighted Shapley values. Games Econ Behav, 1995, 8: 389–405
11 Cheng D Z, Qi H S, Li Z Q. Analysis and Control of Boolean Networks: A Semi-tensor Product Approach. Berlin: Springer, 2011
12 Li F F, Sun J T. Stability and stabilization of Boolean networks with impulsive effects. Syst Control Lett, 2012, 61: 1–5
13 Li H T, Zhao G D, Meng M, et al. A survey on applications of semi-tensor product method in engineering. Sci China Inf Sci, 2018, 61: 010202
14 Meng M, Liu L, Feng G. Stability and l_1 gain analysis of Boolean networks with Markovian jump parameters. IEEE Trans Autom Control, 2017, 62: 4222–4228
15 Lu J Q, Zhong J, Huang C, et al. On pinning controllability of Boolean control networks. IEEE Trans Autom Control, 2016, 61: 1658–1663
16 Zhao G D, Fu S H. Matrix approach to trajectory control of higher-order k-valued logical control networks. IET Control Theory Appl, 2017, 11: 2110–2115
17 Liu Z B, Wang Y Z, Li H T. Two kinds of optimal controls for probabilistic mix-valued logical dynamic networks. Sci China Inf Sci, 2014, 57: 052201
18 Zheng Y T, Li H T, Ding X Y, et al. Stabilization and set stabilization of delayed Boolean control networks based on trajectory stabilization. J Franklin Inst, 2017, 354: 7812–7827
19 Wang Y Z, Zhang C H, Liu Z B. A matrix approach to graph maximum stable set and coloring problems with application to multi-agent systems. Automatica, 2012, 48: 1227–1236
20 Zhong J, Lu J Q, Huang C, et al. Finding graph minimum stable set and core via semi-tensor product approach. Neurocomputing, 2016, 174: 588–596
21 Zhao J T, Chen Z Q, Liu Z X. Modeling and analysis of colored petri net based on the semi-tensor product of matrices. Sci China Inf Sci, 2018, 61: 010205
22 Cheng D Z, Xu T T. Application of STP to cooperative games. In: Proceedings of 10th IEEE International Conference on Control and Automation (ICCA), Hangzhou, 2013. 1680–1685
23 Cheng D Z. On finite potential games. Automatica, 2014, 50: 1793–1801
24 Wang Y H, Liu T, Cheng D Z. From weighted potential game to weighted harmonic game. IET Control Theory Appl, 2017, 11: 2161–2169
25 Cheng D Z, He F H, Qi H S, et al. Modeling, analysis and control of networked evolutionary games. IEEE Trans Autom Control, 2015, 60: 2402–2415
26 Wang Y H, Cheng D Z. Dynamics and stability for a class of evolutionary games with time delays in strategies. Sci China Inf Sci, 2016, 59: 092209
27 Guo P L, Zhang H X, Alsaadi F E, et al. Semi-tensor product method to a class of event-triggered control for finite evolutionary networked games. IET Control Theory Appl, 2017, 11: 2140–2145
28 Branzei R, Dimitrov D, Tijs S. Models in Cooperative Game Theory. Berlin: Springer, 2008
29 Sedgewick R. Permutation generation methods. ACM Comput Surv, 1977, 9: 137–164
30 Rosen K H, Michaels J G, et al. Handbook of Discrete and Combinatorial Mathematics. Boca Raton: CRC Press, 1999