Extracting hidden preferences over partitions in hedonic cooperative games

Athina Georgara, Dimitrios Troullinos, and Georgios Chalkiadakis

School of Electrical and Computer Engineering,
Technical University of Crete, Chania, Greece
{ageorgara, gehalk} @ intelligence.tuc.gr , dtroullinos @ isc.tuc.gr

Abstract. The prevalent assumption in *hedonic games* is that agents are interested solely on the composition of their own coalition. Moreover, agent preferences are usually assumed to be known with certainty. In our work, agents have *hidden* preferences over *partitions*. We first put forward the formal definition of hedonic games in partition function form (PFF-HGs), and extend well-studied classes of hedonic games to this setting. Then we exploit three well-known supervised learning models, *linear regression*, *linear regression with basis functions*, and *feed forward neural networks*, in order to (approximately) extract the unknown hedonic preference relations over partitions. We conduct a systematic evaluation to compare the performance of these models on PFF-HGs; and, in the process, we develop an evaluation metric specifically designed for our problem. Our experimental results confirm the effectiveness of our work.

Keywords: Learning Preferences or Rankings · Cooperative Games · Uncertainty in AI · Applications of Supervised Learning.

1 Introduction

Hedonic games [2] is a class of cooperative games typically used to model settings where agents form coalitions based on their interpersonal relations with others. That is, agents are attracted to coalitions due to their membership, and their "payoff" corresponds to their satisfaction from the collaboration itself. Hedonic games can in fact be modeled as *non-transferable utility games* [7]. In most studies so far, complete information over essential components of such games is assumed. However, under a more realistic point of view, one has to deal with *uncertainty*. In this light, [15] and [9] explore learning methods applied in several classes of hedonic games to discover the underlying hidden aspects of the game.

To the best to our knowledge, all hedonic game papers so far assume that the utility an agent i assigns to a coalition C depends solely on the identities of agents in C [2, 8]; however, the value of a coalition can intuitively be affected not only by its own composition, but also by that of other groups.[1] In other words, it is natural that the satisfaction yielded by a certain coalition differs depending on *coalition structure*. In cooperative game theory, such partition-wide preferences have been studied both in transferable and non-transferable

[1] In [1], in fact, this is mentioned as a potential extension, but it is not studied there.

utility games (games with externalities or in partition function form), but not specifically within the scope of hedonic games [16, 7]. Here we provide the formal definition of hedonic games in partition function form (PFF-HGs), and extend well-studied sub-classes of hedonic games to ones with externalities.

We then proceed to employ supervised learning models such as linear regression, regression with basis functions, and feed-forward neural networks to discover *latent collaboration patterns* in PFF-HGs. In particular, we focus on specific classes of games in PFF-HG, and we exploit their properties to extract their unknown aspects: we consider the preference relation of each agent as the latent aspect of the game. In other words, in our framework agents themselves are unaware of their own preferences; and they *learn* the hidden preferences relation via observing past interactions with others. This is a realistic assumption when considering a vast partition-space: in fact, any competition involving teams or individuals, which can regroup over time and be 'revealed' to the participants, can be captured by the settings we describe in this work. Note that even with a few agents the partition-space is extremely large, e.g., even for 10 agents, we have $B_{10} = \sum_{k=0}^{9} \binom{9}{k} \cdot B_k = 115975$ partitions (B is the *Bell number* [7]).

For instance, consider n companies that establish synergies in order to acquire government contracts. In each competition the companies form different coalitions, and the bundling of the contracts leads to different satisfaction for each company. A series of such competitions provides a learning model with sufficient data that can help an involved company to plan forthcoming collaborations. Another example could be crowdsourcing online platforms, where given a complicated task and a specific time frame, individuals are to form work-groups that compete with each other to solve problems or puzzles; at the end of a task's period, the groups may reform into different coalitions. In such settings, the groupings naturally affect the final outcome, while the formation space is vast and thus own preferences are unknown.

In what follows, in Sec. 2 we formally define the PFF-HG games. In Sec. 3 we present the explicit application of the aforementioned learning models to our problem at hand. In Sec. 4 we describe the game environments within which we employed the learning process. Finally, in Sec. 4.3 we discuss our results, while Sec. 5 concludes this paper.

2 Hedonic Games in Partition Function Form

Let $N = \{1, \cdots, n\}$ be a finite, non-empty set of players of size $|N| = n$. A coalition $S \subseteq N \setminus \{\emptyset\}$ is a non-empty subset of players. The relative to player i coalitional space, consists of all coalitions which contain i, and is defined as $N_i = \{S \subseteq N : i \in S\}$. A *partition* or *coalition structure* is a set of coalitions $\pi = \{S_1, \cdots, S_m\}$ such that for each pair $i, j = 1, \cdots, m$ and $i \neq j$, $S_i \cap S_j = \emptyset$; and $\cup_{i=1,\cdots,m} S_i = N$. We denote with $\pi(i)$ the coalition within π that contains agent i. The set of all partitions of N is denoted as Π. The pair (S, π) is an *embedded coalition*, where $\pi \in \Pi$ is a partition, and $S \in \pi$ is a coalition within partition π. The set of all embedded coalitions over N is denoted with $\mathcal{E}_N$. The

set of the embedded coalitions that contain agent i is denoted with $\mathcal{E}_N(i)$. The class of hedonic games is defined as:

Definition 1 (Hedonic Games [2]). *A hedonic game is given by a pair $G = (N, \succsim)$, where $\succsim$ is a preference profile that specifies for every agent $i \in N$ a reflexive, complete and transitive binary relation $\succsim_i$ on N_i.*

Both transferable (TU) and non-transferable (NTU) utility games have been expressed in partition function form (PFF). Hedonic games constitute a special class of NTU games:

Definition 2 (NTU games in Partition Function Form [14]). *A coalitional game in partition function form (PFF) with non-transferable utility (NTU) is defined by a pair $\langle N, V\rangle$, where N is the set of players, and V is a mapping such that for every $\pi \in \Pi$ and every coalition $S \subseteq N$, $S \in \pi$, $V(S, \pi)$ is a closed convex subset of $\mathbb{R}^{|S|}$ that contains the payoff vector that players in S can achieve. Alternatively, if we consider a payoff vector in $\mathbb{R}^n$ for every coalition $S \subseteq N$ (let for any $i \notin S$ the corresponding payoff be 0 or $-\infty$), then V can be viewed as a mapping $V : \mathcal{E}_N \to \mathbb{R}^n$ that assigns to n-vector of real numbers to each embedded coalition (S, π).*

We now provide the corresponding definition for *hedonic games in partition function form*, and extend common hedonic game classes to this setting.

Definition 3. *(PFF-HGs) A hedonic game (HG) in partition function form (PFF) is defined by a pair $\langle N, \succsim\rangle$, where N is the set of players, and $\succsim = \{\succsim^{\pi_1}, \cdots, \succsim^{\pi_m}\}$ with $|\Pi| = m$; and for all $\pi_j \in \Pi$ $\succsim^{\pi_j} = \{\succsim_1^{\pi_j}, \cdots, \succsim_n^{\pi_j}\}$, and each $\succsim_i^{\pi_j} \subseteq N_i \times N_i$ is a complete, reflexive and transitive preference relation describing agent i's preferences over coalitions it can participate in when π_j is in place.*

2.1 Additively Separable Hedonic Games (ASHGs)

In ASHGs [8], an agent's utility for a given coalition is the sum of the utilities it assigns to other members of that coalition. Formally, each agent $i \in N$ assigns to each agent $j \in N$ a value $b_i^j \in \mathbb{R}$,[2] and the utility of coalition S is defined as $v_i(S) = \sum_{j \in S} b_i^j$.

Generalizing ASHGs to partition function form, each agent assigns a value to any agent within each partition $\pi \in \Pi$, i.e, agent i assigns the value $b_i^j(\pi)$ to agent j when partition π is formed. Thus, the utility of embedded coalition (S, π) is now defined as $v_i(S, \pi) = \sum_{j \in S} b_i^j(\pi)$. As such, it is more natural to model ASHGs like NTU games due to their properties. Therefore, there is a mapping $V : \mathcal{E}_N \to \mathbb{R}$ such that for every embedded coalition there is an n-vector of reals:

$$V(S, \pi) = \begin{bmatrix} v_1(S, \pi) \\ v_2(S, \pi) \\ \vdots \\ v_n(S, \pi) \end{bmatrix} = \begin{bmatrix} \sum_{j \in S} b_1^j(\pi) \\ \sum_{j \in S} b_2^j(\pi) \\ \vdots \\ \sum_{j \in S} b_n^j(\pi) \end{bmatrix}$$

[2] The agents assign a zero value to themselves, i.e., $b_i^i = 0$.

Given this, each agent forms a preference relation that refers to embedded coalitions (rather than coalitions), and therefore to partitions. This preference relation is as follows: agent i prefers embedded coalition (S, π) to (T, π'), $(S, \pi) \succsim_i (T, \pi')$, if and only if $v_i(S, \pi) \geq v_i(T, \pi')$, even if S and T consist of exactly the same set of agents.[3]

2.2 Boolean Hedonic Games (BHGs)

Boolean hedonic games provide a concise representation of hedonic games with dichotomous preference relations [1]. According to the dichotomous preferences model, each agent i can partition $N_i = \{S \subseteq N \setminus \{\emptyset\} : i \in S\}$ into two disjoint sets N_i^+ and N_i^-; and i strictly prefers all coalitions in N_i^+ to those in N_i^-, and is indifferent about coalitions in the same set. In boolean hedonic games, each agent i, instead of explicitly enumerating the preference relation that leads to dichotomous preferences, defines a logic formula γ_i that intuitively represents its goal of being with preferred partners; and i is satisfied if its goal is achieved, or dissatisfied otherwise. This formula can be of any form of a propositional logic language, but γ_i in [1] involves only propositional variables *relative* to agent i, that is, denoting agents i wants or does not want to be grouped with.

Expanding BHGs to partition function form, the key idea is to partition the Π space into two disjoint sets P_i^+ and P_i^-, i.e. into a set with the partitions agent i prefers, and a set with the partitions i does not. In its generality, the use of propositional logic formulae allows us to have a compact representation, but γ_i in [1] was meant to capture i's preferences regarding the composition of its coalition only. We extend into PFF by introducing a specific form for γ_i, which is as follows: each γ_i is *not* restricted to variables relative to agent i, but consists of multiple pairs $\langle Incl_i, \overline{Incl_i} \rangle$ connected via the logical connective *or* ($\vee$). $\langle Incl_i, \overline{Incl_i} \rangle$ is a pair of two *disjoint* sets of subcoalitions. Now, each pair is interpreted as follows: $Incl_i$ is a set of "must-include" subsets of coalitions, while $\overline{Incl_i}$ is a set of "must-not-include" subsets of coalitions. In words, the set $Incl_i$ represents desirable patterns of collaborations that a preferable to i partition must contain; while the set $\overline{Incl_i}$ indicates cooperation among agents that must be excluded from a preferable partition. Thus, a partition π satisfies the pair $\langle Incl_i, \overline{Incl_i} \rangle$ if:

- $\forall c \in Incl_i \ \exists S \in \pi : c \subseteq S$; **and**
- $\forall c \in \overline{Incl_i} \ \nexists S \in \pi : c \subseteq S$

that is, for every desirable pattern $c \in Incl_i$ there is a coalition $S \in \pi$ that contains c (i.e., $c \subseteq S$); while for each unwanted pattern $c \in \overline{Incl_i}$ there is no coalition $S \in \pi$ such that c is contained in S. In general, a formula γ_i is of the form: $\gamma_i = \langle Incl_{i,1}, \overline{Incl_{i,1}} \rangle \vee \langle Incl_{i,2}, \overline{Incl_{i,2}} \rangle \vee \cdots \vee \langle Incl_{i,p}, \overline{Incl_{i,p}} \rangle$, and a partition π satisfies γ_i (we write $\pi \models \gamma_i$) if there is a pair $\langle Incl_{i,j}, \overline{Incl_{i,j}} \rangle$ such that π satisfies $\langle Incl_{i,j}, \overline{Incl_{i,j}} \rangle$. Therefore, the partition space Π is the disjoint union of the sets: $P_i^+ = \{\pi \in \Pi : \pi \models \gamma_i\}$ and $P_i^- = \{\pi \in \Pi : \pi \not\models \gamma_i\}$.

[3] Notice, that by simply changing the sum operator with max, min, or average, we can similarly express the $\mathcal{B} - Games$, $\mathcal{W} - Games$, and FHGs [2] in PFF, respectively.

3 Learning Models

Here we discuss learning models that we applied on PFF-HGs.

3.1 Linear Regression Model

The linear regression model (LRM) [6] is a simple but powerful data analysis tool. An LRM attempts to find the best *line* fitting the input observations ($\mathbf{x}$) and their target values ($\mathbf{t}$). Each observation can be an L-dimensional vector, that abbreviates multiple parameters. During the training phase, the LRM computes a weight vector $\mathbf{w} = (\mathbf{x}^{\mathrm{T}}\mathbf{x})^{-1}\mathbf{x}^{\mathrm{T}}\mathbf{t}$; while during the testing phase, the model compares each approximated value $\hat{y}_r = w_0 + \sum_{l=1}^{L} w_l \cdot x_{r,l}$ with its corresponding target value t_r.

We let each agent $i \in N(|N| = n)$ train and maintain its own learning model. Given a hedonic game G in PFF, an observation is a pair (partition, target value), encoded as follows: we use $\binom{n}{2}$ boolean variables, with each one indicating whether an unordered pair of agents co-exist in the same coalition within the given partition. That is, for each unordered pair (i, j) we use the indicator function $\mathbb{1}_{i\in\pi(j)} = 1$ if agents i, j are in the same coalition, and $\mathbb{1}_{i\in\pi(j)} = 0$ otherwise.[4] Thus, a partition π_k is encoded as $\mathbf{x}_k$ having $\binom{n}{2}$ elements as:

$$\mathbf{x_k} = [\underbrace{\mathbb{1}_{ag_2\in\pi(ag_1)}, \mathbb{1}_{ag_3\in\pi(ag_1)}, \cdots, \mathbb{1}_{ag_n\in\pi(ag_1)}}_{n-1 \text{ elements}}, \underbrace{\mathbb{1}_{ag_3\in\pi(ag_2)}, \mathbb{1}_{ag_4\in\pi(ag_2)}, \cdots, \mathbb{1}_{ag_n\in\pi(ag_2)}}_{n-2 \text{ elements}},$$
$$\underbrace{\mathbb{1}_{ag_4\in\pi(ag_3)}, \mathbb{1}_{ag_5\in\pi(ag_3)}, \cdots, \mathbb{1}_{ag_n\in\pi(ag_3)}}_{n-3 \text{ elements}}, \quad \cdots\cdots\cdots\cdots,$$
$$\underbrace{\mathbb{1}_{ag_{i+1}\in\pi(ag_i)}, \mathbb{1}_{ag_{i+2}\in\pi(ag_i)}, \cdots, \mathbb{1}_{ag_n\in\pi(ag_i)}}_{n-i \text{ elements}}, \quad \cdots\cdots\cdots\cdots,$$
$$\underbrace{\mathbb{1}_{ag_{n-1}\in\pi(ag_{n-2})}, \mathbb{1}_{ag_n\in\pi(ag_{n-2})}}_{2 \text{ elements}}, \mathbb{1}_{ag_n\in\pi(ag_{n-1})}]$$

As mentioned above, each agent trains its own model to learn its preferences, considering that each agent's target value for a given partition can be different. Training and testing samples contain partitions (π_k) encoded as $\mathbf{x}_k$, along with the corresponding target value (t_k). For ASHGs, this value is equal to $t_k = v_i(S, \pi_k)$, whereas for BHGs, t_k will have a positive value $+c$, $(c > 0)$ if $\pi_k \in P_i^+$ or a negative value $-c$ if $\pi_k \in P_i^-$. In essence, learning an agent's preferences in BHGs can be described as a classification problem, and is examined as such; while ASHGs is a straightforward function approximation problem.

3.2 Linear Regression Model with Basis Functions

The linear regression model provides us with a linear function of the form $t_i \sim \alpha + \beta x_i$. However, the data may not be linear, and thus a line is not

[4] We remind the reader that $\pi(j) = S \in \pi : j \in S$.

an appropriate function to use as predictor. Instead, we can empower the LRM by using M basis functions; i.e. a number of non-linear functions that help us to acquire a more appropriate predictor. The weight vector is now defined according to this set of basis functions, and $\mathbf{w}$ is computed as: $\mathbf{w} = (\boldsymbol{\Phi}^T \cdot \boldsymbol{\Phi})^{-1} \cdot \boldsymbol{\Phi}^T \cdot \mathbf{t}$, where $\boldsymbol{\Phi} = \begin{bmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & & \ddots & \vdots \\ \phi_0(\mathbf{x}_K) & \phi_1(\mathbf{x}_K) & \cdots & \phi_{M-1}(\mathbf{x}_K) \end{bmatrix} \in \mathbb{R}^{K \times M}$. The number M of basis functions, along with their form, depend essentially on the problem at hand; here we determine M with the TPE method described in Sec. 3.3. In our approach, we chose to work with *Gaussian Basis Functions*. These are of the form [6]:

$$\phi_i(\mathbf{x}) = exp\Big\{ - \frac{\|\mathbf{x} - \mu_i\|^2}{2\sigma^2} \Big\} \tag{2}$$

where μ_i reflects the location of each ϕ_i in the L-dimensional space, and σ the scale of each value, which is common to all ϕ_i. Now, each basis function ϕ_i is an exponential, related to a center vector $\mu_i \in \mathbb{R}^{\|X\|}$, and a standard deviation $\sigma \in \mathbb{R}^+$. For computing the center vector μ_i of each ϕ_i, we use the well-known *k-means* algorithm [12], while σ is the same for all ϕ_i.

3.3 Feed-Forward Neural Network (NN)

The aforementioned learning models are not in general appropriate for modelling complex preferences with great accuracy. Feed-Forward Neural Networks [6, 10] are better at that task, and thus we also employ one here.

In each hidden layer, a non-linear activator is used, otherwise it would act similarly to an LR model. Depending on the PFF-HG class, we let the output layer have a different activator. That is, in ASHGs, a regression model is needed to approximate the function $t_k = v_i(S, \pi_k) \in \mathbb{R}$, so the output layer activator must be linear. In the case of BHGs, essentially we have a classification problem, thus we use a sigmoid activator at the output layer. By using a sigmoid function, the resulting target values lie in $\{0, 1\}$; thus we use the convention that when the target value of a given sample π_k is $t_k = 1$, it means that $\pi_k \in N_i^+$, and when $t_k = 0$ we have that $\pi_k \in N_i^-$.

The performance of an NN depends on the choice of some hyperparameters such as the optimization method, the number of nodes per layer, the activator function in each layer, etc.; while the chosen set of hyperparameters is highly dependent on the complexity of the to-be-learned preferences, i.e., the problem at hand. In this work, we use as an optimization method the *ADAM optimization* algorithm [3]. ADAM attempts to combine the benefits of two different variations of the stochastic gradient descent method, namely Adaptive Gradient Descent (AdaGrad) and the Root Mean Square Propagation (RMSprop) [13]. AdaGrad uses a different learning rate for each parameter to improve performance for sparse gradients, while in RMSprop and AdaDelta the learning rate of each parameter relies on a decaying average of recent gradients. The choice of the

ADAM optimizer was made empirically, following a series of experiments with instances of our problem. This showed that ADAM outperformed AdaGrad, RMSprop and AdaDelta [13]. We focused on the above methods, since adaptive learning systems are generally suggested for sparse data, as discussed in [13]. The sparsity of the data is clearly indicated by the form of our observations' representation (see Sec. 3.1).

Having selected the optimization method, we let our architecture self-tune the other hyperparameters of the network by using the Tree-structured Parzen Estimator (TPE) [5], an algorithm based on Bayesian optimization with the use of *hyperopt library* for python [4]. In general, hyperparameter optimizers attempt to find a value h_i that minimizes a function $f(h)$, i.e. $\text{argmin}_h(f(h))$. In our apporach, f represents the loss function of the neural network, since this is the quantity we want to minimize. For a predefined number of steps, the algorithm produces sets of hyperparameters h_i, that are used to construct a neural network, which is then trained and tested given the input data. At the end of this process, our architecture yields the best set of hyperparameters h^*, along with the trained neural network that uses these h^*. The TPE allows us to optimize hyperparameters that are structurally dependent: for instance, if we set as a hyperparameter the number of hidden layers, we first optimize the number of layers, and then the hyperparameters for each layer. Fig. 1 shows the architecture of the model for selecting the best hyperparameters for the neural network. In Sec. 4.2 we introduce an evaluation metric, which is used as the function f.

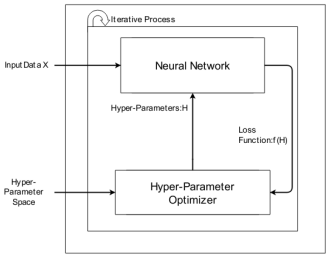


Fig. 1: Architecture for selecting network hyperparameters.

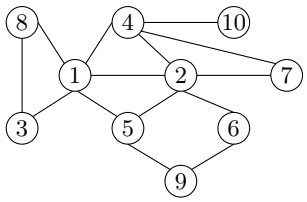


Fig. 2: Social graph g example

4 Experimental Results

Here we first describe the games' environment, i.e., how we generated the synthetic data used for our experiments. Then, we present the evaluation metrics of our work. The coding was in Python3; the experiments ran on a PC with an i7@2.7GHz processor and 8GB of RAM.

4.1 Games environment setup

The preference relations in hedonic games may be arbitrary. In the classes mentioned above, the preferences depend either on the values b_i^j with a known operator (summation, maximization, etc.), or on formulae. The values b_i^j may be arbitrary, follow some a priori known distribution, or follow some specific function. Here, we suggest that the values b_i^j are derived from a graph, reminiscent of a social network,[5] that describes the "communication links" between the agents.

Specifically, the value i assigns to j within partition π, comes from the graph structure. That is, $b_i^j(\pi)$ is related to the following three properties:

1. the distance between agent i and j; and
2. the distances between agent i and all coalitions in π.
3. the distances between agent j and all coalitions in π.

Intuitively, the distance between i and j, denoted as $\text{dist}_g(i,j)$, represents a measurable metric for the expression "how close friends are i and j". Property 1 would have been enough in hedonic games with no externalities. However, when we study the games in partition function form, each agent values differently coalitions and agents depending on the formed coalition structure. Thus, we need to capture the partition's form. The distance between an agent i and a coalition S in π, intuitively represents i's 'motivation' to deviate in another existing coalition. Therefore, property 2 compensates i for collaborating with j instead of another set of people; while property 3 'honours' j for working with i.

The distance between two agents, $\text{dist}_g(i,j)$, is defined as the shortest path that connects these agents; and the distance between an agent i and a coalition S is $\text{dist}_g(i,S) = \sum_{j \in S} \frac{\text{dist}_g(i,j)}{|S|}$. Hence, the value i assigns to j is defined as:

$$b_i^j(\pi|g) = \kappa_i \cdot \frac{1}{\text{dist}_g(i,j)} + \lambda_i \cdot \sum_{S \in \pi} \text{dist}_g(i,S) + \xi_i \cdot \sum_{S \in \pi} \frac{1}{\text{dist}_g(j,S)},$$

where g is the graph, and κ_i, λ_i and ξ_i are constants related to agent i. In words, term $\lambda_i \cdot \sum_{S \in \pi} \text{dist}_g(i,S)$ captures i's 'cost' to move to a different coalition and attributes a small/large amount for coalitions close/faraway to her. While term $\xi_i \cdot \sum_{S \in \pi} \frac{1}{\text{dist}_g(j,S)}$ expresses i's degree of honour towards j, and attributes a small/large amount for coalitions faraway/close to j.

Note that each agent may consider a completely different social network than its peers. In other words, each i perceives differently the collaboration patterns among players. Having defined values b_i^j in this way, we may therefore use any of the above suggested utility functions, e.g. summation, boolean, etc., to determine the coalitions' and partitions' utility.

Game parameters In all games we have $n = [5, 10, 20, 50]$ agents. For the ASHGs in PFF: in each game we constructed a social graph with n nodes and $e \sim \mathcal{U}(n-1, \frac{n \cdot (n-1)}{2})$ number of edges (uniformly chosen between the minimum number of edges that allow us to have a connected graph, and the number of

[5] We use the term 'social network' to simply refer to a graph that produces the personal values b_i^j.

	Graph			Formulae	
n	κ, λ, ξ	$\|edges\|$	$edge_{weight}$	$\#\langle Incl_i, \overline{Incl_i}\rangle/agent$	$\#agents/\langle Incl_i, \overline{Incl_i}\rangle$
5	$\sim \mathcal{U}(0, 10)$	$\sim \mathcal{U}(4, 10)$	$\mathcal{U}(1, 5)$	$\mathcal{U}(2, 3)$	$\mathcal{U}(1, 4)$
10	$\sim \mathcal{U}(0, 10)$	$\sim \mathcal{U}(9, 45)$	$\mathcal{U}(1, 5)$	$\mathcal{U}(4, 7)$	$\mathcal{U}(4, 6)$
20	$\sim \mathcal{U}(0, 10)$	$\sim \mathcal{U}(19, 190)$	$\mathcal{U}(1, 5)$	$\mathcal{U}(9, 11)$	$\mathcal{U}(5, 7)$
50	$\sim \mathcal{U}(0, 10)$	$\sim \mathcal{U}(49, 1225)$	$\mathcal{U}(1, 5)$	$\mathcal{U}(12, 15)$	$\mathcal{U}(8, 12)$

Table 1: Game environment parameters

edges that results a fully connected graph); each edge has a weight $w_{edge} \sim \mathcal{U}(1, 5)$. For the BGs in PFF: in each game, we use uniform distributions to construct a formula $\boldsymbol{\gamma}_i$ consisting of multiple pairs $\langle Incl_i, \overline{Incl_i}\rangle$ for each agent $i \in N$, depending on the number of agents in the game; and to define the number of agents involved in each pair, as shown in Table 1.

4.2 Evaluation metrics

In order to evaluate our experimental results, it is essential to determine the nature of the desirable outcomes. We work with hedonic games, a class of cooperative games which, unlike others, possesses some particular properties. Specifically, in such games, we care little about the actual coalitions' utilities, since our interest lies mainly on the preference relations formed. The modelling of the games studied within the scope of this paper (i.e., the interpretation of a game instance into input data for a learning model), allows us to extract the desired preferences. Thus, a question arises: do we care to learn the best function $\hat{u}_i(S, \pi)$ that resembles the actual $u_i(S, \pi)$; or do we desire a $\hat{u}_i(S, \pi)$ that encodes a preference relation best matching the actual one?
To be more accurate, we distinguish the following two evaluation metrics:

- the *Root Mean Square Error* (RMSE) between the predicted utility functions $\hat{u}_i(S, \pi)$, and the true function $u_i(S, \pi)$
- the *Qualitative Proximity* (QP) between the ordering described by $\hat{u}_i(S, \pi)$ and the true preference relation.

The first metric is straightforward, $\text{RMSE}(u_i, \hat{u_i}) = \sqrt{\frac{1}{|D|}\sum_{(S,\pi)\in D}\left(u_i(S, \pi) - \hat{u}_i(S, \pi)\right)^2}$, where D is the collection of testing data encoded as described in Sec. 3.1, and $|D|$ is the size of the collection D. The second metric, however, is more interesting. This metric indicates that even if the predicted $\hat{u}_i$ differs significantly from the true u_i in actual values, they are still equivalent. That is, functions $\hat{u}_i$ and u_i encode in the same way the preference relation between any two embedded coalitions (S, π) and (T, π'), e.g. both $(S, \pi) \succsim_i^{u_i} (T, \pi')$ and $(S, \pi) \succsim_i^{\hat{u}_i} (T, \pi')$ hold. QP in fact could be thought of as a variant of *Kendall Tau* metric [11].

Therefore, when a utility function of an agent i is "learned" based on the training data, we extract a preference relation, $\succsim_i^{\hat{u}_i}$, over partitions. Then we can measure the *percentage of equivalence* between u_i and $\hat{u}_i$ by counting the average of pairwise relations that are identically encoded by u_i and $\hat{u}_i$. Thus, we define Qualitative Proximity as follows:

$$\text{QP}(u_i, \hat{u_i}) = \frac{\sum_{(S,\pi),(T,\pi')\in D} \text{CHK}\big(u_i, \hat{u}_i, (S, \pi), (T, \pi')\big)}{|D|(|D| - 1)/2}, \text{ where}$$

$$\mathrm{CHK}\big(u_i, \hat{u}_i, (S,\pi), (T,\pi')\big) = \begin{cases} 1 & \text{if } \big((S,\pi) \succsim_i^{u_i} (T,\pi') \\ & \quad \wedge\ (S,\pi) \succsim_i^{\hat{u}_i} (T,\pi')\big) \\ 0 & \text{otherwise} \end{cases}.$$

One step further, we introduce a single metric that combines both RMSE and QP. According to this metric:

- the lower the error $\mathrm{RMSE}(u_i, \hat{u_i})$, the better $\hat{u}_i$ fits u_i
- the higher the $\mathrm{QP}(u_i, \hat{u_i})$, the better $\succsim_i^{\hat{u}_i}$ matches $\succsim_i^{u_i}$.

Thus, our *Overall Preference Accuracy (OPA)* metric can be defined as follows:

$$\mathrm{OPA}(u_i, \hat{u}_i) = \frac{\mathrm{QP}(u_i, \hat{u}_i)}{\epsilon + \mathrm{RMSE}_{\mathrm{nrm}}(u_i, \hat{u}_i)},$$

where $\mathrm{RMSE}_{\mathrm{nrm}}$ is the normalized RMSE regarding the test samples values' range. This normalized RMSE allows us to compare the performance in games with different value ranges. The intuitive interpretation of the OPA metric is that, we highly value the contribution of QP metric–since this actually measures the similarity of the original preference relation to the approximated one–but also take into some consideration the RMSE between the original and the approximated function. In order to avoid division with zero, we add a small positive $\epsilon = 10^{-5}$ to the denominator.

As mentioned above, we use OPA in the TPE for the hyperparameter selection. Specifically, we set the loss function TPE attempts to minimize to $f(h) = \frac{1}{\mathrm{OPA}(u_i, \hat{u}_i)}$, where u_i is derived from the input (test) data, and the estimated $\hat{u}_i$ is related to the hyperparameters h for the neural network model.

4.3 Results

	5			10			20			50		
	LRM	LRMRBF	NN	LRM	LRMRBF	NN	LRM	LRMRBF	NN	LRM	LRMRBF	NN
ASHGs	0.05sec	1.2sec	14sec	2.4sec	17sec	1.7min	10sec	1.7min	8.3min	3.2min	25.3min	5.5hr
BHGs	0.04sec	1sec	45sec	1.3sec	12sec	2.9min	5.5sec	1.2min	8.4min	1.7min	30.6min	2.5hr

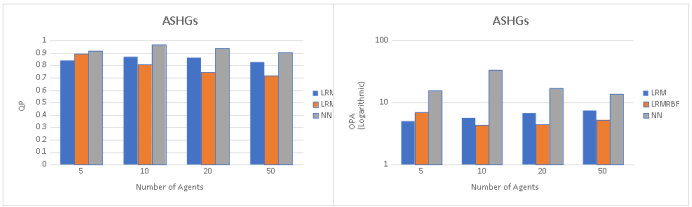
Table 2: Approximate time needed per game for training & testing.

n	Partition samples		
	Training	Testing	Validation
5	200	500	200
10	2000	5000	2000
20	5000	10000	5000
50	20000	40000	10000

Table 3: Samples per setting

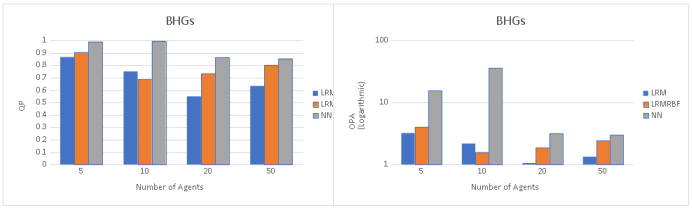
We conducted a series of experiments for our evaluation, simulating both ASHGs and BHGs games. Table 3 shows the number of observation samples used in each setting (depending on the number of agents). Note that *Validation* refers to the process of optimizing the model's hyperparameters. For each (*type of game, number of agents*) setting, we trained each learning model for 5 game examples. In Fig. 3 we show the average scores of both OPA and QP metrics

each learning model yielded per setting (*type of game, number of agents*). As we can see, NN models outperform LRM and LRMRBF, both in ASHGs and BHGs, especially as the number of agents increases. This is, in fact, an anticipated result since the hyper-parameters optimization makes our NN models more adaptive to the problem. However, the larger the number of agents is, the more computationally expensive the NN model is. This results from the hidden layers having many fully connected nodes. The number of layers is 1 or 2, while the nodes per layer are in $[\frac{n}{2}, \frac{n\cdot(n-1)}{2}]$, selected by TPE optimizer.



(a) ASHGs : QP

(b) ASHGs : OPA



(c) BHGs : QP

(d) BHGs : OPA

Fig. 3: QP and OPA scores (average over 5 games).

In BHGs, the complexity of our hidden preferences mainly depends on the number of different pairs $\langle Incl, \overline{Incl}\rangle$ each agent has, and on conflicts regarding the preference for a specific agent j (j may be "wanted" as part of a specific subcoalition, but not of some other). In our experiments we increased the "intricacy" of the preferences, as the number of agents increases (see Table 2). Thus, we observe that the LRM, in BHGs, lacks scalability; i.e., LRM is incapable to approximate highly "complex" preferences. However, for a small number of agents (i.e. 5), the fact that the formulae are considerably simple results to an average $\sim 85\%$ on QP; this, along with the fact that the NN is computationally expensive gives rise to a trade off between the two models–see the approximate time each model needs in Table 2. For ASHGs, due to their nature, both LRM and LRMRBF are very effective in terms of QP, and well-behaved regarding scalability. Once again NN outperforms the other models, however, the high computational load of this model makes the simple and more inexpensive models very appealing. Between the LRM and LRMRBF, the former is found to be more consistent for settings with many players. Note that all three learning methods exhibit performance ranging from adequate to extremely successful, operating with a small number of samples in huge partition space. For example, for 20 agents, we have $10^{13.71372}$ partitions in total, and we feed the models only with 5000 training samples, i.e., with the $9.66 \cdot 10^{-9}\%$ of the total space.

5 Conclusion and Future Work

In this paper, we put forward the formal definition of PFF-HGs, and extended well studied classes of HGs to partition function form. Then, we employed three learning models to approximate the hidden preference relations. Finally, we conducted a systematic evaluation that verified the effectiveness of our approach. As future work, we intend to test the learning process in other classes of hedonic games, such as $\mathcal{B}/\mathcal{W} - Games$, and Fractional HGs [2] in PFF.

References

1. Aziz, H., Harrenstein, P., Lang, J., Wooldridge, M.: Boolean hedonic games. In: Proc. of the Fifteenth International Conference on Principles of Knowledge Representation and Reasoning. pp. 166–175. KR'16, AAAI Press (2016)
2. Aziz, H., Savani, R., Moulin, H.: Hedonic games. In: Brandt, F., Conitzer, V., Endriss, U., Lang, J., Procaccia, A.D. (eds.) Handbook of Computational Social Choice, pp. 356–376. Cambridge University Press (2016)
3. Ba, J., Kingma, D.: Adam: A method for stochastic optimization. In: Proc. of the 3rd International Conference on Learning Representations (ICLR-15) (2015)
4. Bergstra, J., Yamins, D., Cox, D.D.: Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures (2013)
5. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Proc. of NIPS-2011. pp. 2546–2554 (2011)
6. Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg (2006)
7. Chalkiadakis, G., Elkind, E., Wooldridge, M.: Computational Aspects of Cooperative Game Theory (Synthesis Lectures on Artificial Inetlligence and Machine Learning). Morgan & Claypool Publishers, 1st edn. (2011)
8. Elkind, E., Wooldridge, M.: Hedonic coalition nets. In: Proc. of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 1. pp. 417–424. AAMAS '09 (2009)
9. Georgara, A., Ntiniakou, T., Chalkiadakis, G.: Learning hedonic games via probabilistic topic modeling. In: Proc. of the 16th European Conference on Multi-Agent Systems (EUMAS-2018) (2018)
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016)
11. Kendall, M.: Rank correlation methods. Griffin, London (1948)
12. MacQueen, J., et al.: Some methods for classification and analysis of multivariate observations. In: Proc. of the fifth Berkeley symposium on mathematical statistics and probability. vol. 1, pp. 281–297. Oakland, CA, USA (1967)
13. Ruder, S.: An overview of gradient descent optimization algorithms (2016)
14. Saad, W., Han, Z., Zheng, R., Hjorungnes, A., Basar, T., Poor, H.V.: Coalitional games in partition form for joint spectrum sensing and access in cognitive radio networks. IEEE Journal of Selected Topics in Signal Processing pp. 195–209 (2012)
15. Sliwinski, J., Zick, Y.: Learning hedonic games. In: Proc. of the 26th IJCAI-17. pp. 2730–2736 (2017)
16. Thrall, R.M., Lucas, W.F.: N-person games in partition function form. Naval Research Logistics Quarterly **10**(1), 281–298 (1963)