Optimizing Game Theory Algorithms for Graph-Based Games

Giovanni Beraldi[1] und Andrea Cuteri[2]

Abstract: Many solution concepts in Game Theory have been proved to be intractable in the general case. However, in more restrictive cases several results of tractability have been found [GLS17] . In this report we do not show tractability results, but we show that pushing the rules of a game can be beneficial for the computation of hard solution concepts. In particular, we consider Shapley Value for coalitional games and Nash Equilibria for the non-cooperative setting. Results show that we succeeded in speeding up Shapley value and Nah Equilibria computation by enforcing some key properties of the considered game.

Keywords: Shapley Value, Nash Equilibria, Tree Decomposition, Mechanism Design

1 Introduction

In this project we consider four different settings. All the settings are based on games represented over a graphs. The nodes of the graph form the set of players N while the edges $E \in N \times N$ express relation between nodes.
The first point of the project consists fairly distributing the worth obtained by forming a coalition that is capable of connecting the special nodes of the graph s and t. The solution concept we adopt for this point is the Shapley Value. The solution is discussed in section 2.
The second point defines a non-cooperative game in which each agent might decide either to cooperate in the formation of the path or to defect. In section 2 we show how we implement they payoffs for the agents and the computation of Nash Equilibria.
The third point asks to provide the same solution to point 2 that exploits tree decomposition. We discuss the solution in section 4.
In the fourth point the objective is to define the maximum weighted path between s and t, with the weights defined by declarations from players, and enforce the agents to truthfully reporth their value. In section 5 we discuss the implementation.

2 Shapley Value

The setting is a Characteristic fuction Game(CFG) which is a subset of Cooperative Games. The game consists of a set N of players which may form coalitons $C \in N$. In CFG games, payoffs of coalitions are determined only by the components of the

1 University, of, Calabria, gvnberaldi@gmail.com
2 University, of, Calabria, cuteri.andrea@gmail.com

coalition. In our case the value of any given coalition $C \in N$ can be either 0 or 100 and it is determined by a function that checks whether the path forms between the special nodes s and t of the graph. The characteristic function of this game is of the type $f : \mathcal{P}(N) \longrightarrow \{0, 100\}$ The setting is super-additive since for whatever coalitions $C_1, C_2 \in N \; s.t. \; C_1 \cap C_2 = \emptyset \;\; v(C_1 \cup C_2) \geq v(C_1) + v(C_2)$ (in our case it is equal if at least one of the two coalitions form the path).
The solution consists of two candidates. The first is a trivial solution that represents the full characteristic function. The second solution represents only the winning coalitions - those that have a value associated of 100. The reason why this approach works is quite simple and is described directly in the rules of the game. Let us recall the formula of the shapley value $\phi_i = \sum_{C \subseteq N} \frac{(N-S)! * (S-1)!}{N!} * (v(C) - v(C \setminus \{i\}))$.
In the previous formula $(v(C) - v(C \setminus \{i\}))$ represents the marginal contribution of player i in coalition C. It is easy to notice from the rules of the game that a player has marginal contribution different from 0 in coalition C iff he is a pivotal player for that coalition. This means that only after adding player i to coalition C the path forms. Now, an easy yet powerful trick is to represent the CF by storing only winning coalitions and setting to 0 the marginal contribution for whatever player i in C that is non pivotal(which in terms of code translates into checking if the $v(C \setminus \{i\})$ is stored or not). The case in which both C and $C \setminus \{i\}$ comes for free since the difference of their associated values is 0. An even smarter solution could be representing in the CF only the minimal subsets of nodes that form the path and then count for every player in how many super sets of the minimal subsets they would have a marginal contribution that is not zero, but for the sake of simplicity we just considered the one we have described before. In section 6 we highlight the obtained results.

3 Nash Equilibria

The setting is a non-cooperative or strategic game with an arbitrary number of players $N = \{1, 2, \cdots, n\}$ represented by the nodes of a graph. Each agent can choose either to cooperate or to defect. The objective is to find an equilibrium given that no node wants to cooperate if more that two of its neighbours are cooperating. Nodes that cooperate may or may not form a path between s and t.
Given a strategy profile $S =< S_1, S_2, \cdots S_n >$ where $S_i \in \{C, D\}$ the payoffs are constructed as follows. A node that cooperates has payoff:

- 1 if no more that two of its neighbours are cooperating in S and the path forms
- -1 if three or more of its neighbours are cooperating in S
- 0 if no more that two of its neighbours are cooperating in S but the path does not form

A node that defects has payoff:

- -0.5 in any case

Payoffs are represented using a dictionary in which keys represent players that are cooperating in a given strategy profile S while the values are lists of size n where n is the cardinality of nodes excluding s and t. Consider a graph with 4 nodes numbered 1..4 except s and t. The strategy profile $S =< C, C, C, D >$ in which the nodes 1 to 3 cooperate and 4 defects is represented as the frozenset $1, 2, 3$. For what concerns the payoffs, they are represented via a list of size 4 in which the ith component describes the payoff of the ith node of the graph in the strategy profile given by the key of the dict entry. Such a representation is possible since the available actions for all the players are just two and hence we can interpret a missing player in a frozenset as a defecting player. Also for this point we provide two solution. The naive one consists in enumerating all the strategy profile and checking if each of them is an equilibrium. On the other hand, the slightly smarter solution tries to cut the number of strategy profiles that are to be checked since it considers nodes with at most two neighbours as always cooperating. The idea is based on a very simple consideration that no equilibrium can exist when this type of players are defecting. Note that if all the nodes are always-cooperating nodes, a trivial NNash Equilibrium is the strategy profile in which all players cooperate(no need for enumeration or checks). In section 6 we provide more details about the proposed solution and we discuss the results.

4 Tree Decomposition

As an extension to the Nash Equilibria computation, we try to use dynamic programming for computing Nash Equilibria. However, there is an important obstacle in this game settings that makes it hard to provide an easy and efficient solution of the problem by applying dynamic programming. As already discussed in Section 3, we define payoffs according to connectivity between the special nodes s and t in such a way that we partially promote cooperation. For this reason we can not solve subproblems represented by bags in a local fashion like we do with the Maximum Independent Set problem. Indeed, it is necessary for us to check the entire payoffs at the leaves-level in order to fix and then propagate candidate Nash Equilibria.
From leaves to root the number of strategy profiles to be checked shrinks, but still, from some empirical evaluation, we noticed that the Tree Decomposition like solution we provide does not scale. Another solution that only considers the at most two neighbours cooperating and not the connectivity between s and t would probably scale and lead to much better performances w.r.t. to the provided one, but for reasons related to time we are not able to provide it.

5 Mechanism Design

The goal of this part of the project was to define a mechanism that would incentive the nodes inside the graph except s and t to truthfully report their internal utility. The obective of the game was to create the maximum weighted path between s and t using the declaration

of the nodes as weights. Therefore a node is selected if they are in the maximum weighted path between s and t. Any rational agent in this setting would increase their declaration with the hope that said declaration is high enough and they are selected by the game. To address this issue, we design a VCG mechanism so that each player finds declaring their real utility the best option.

6 Experiments

To assess performances of our implementations we conducted some bernchmarks for both Shapley Value and Nash Equilibria computation. All experiments have been ran inside the CPU environment of Google Colab and timings were taken using the time module of Python. The results are shown in the following subsections. For mechanism design we simply show how reporting the truth is the best option a player can do.

6.1 Shapley Value

We consider two categories of graphs: dense graphs and sparse graphs. The threshold between the two types is set at 0.5 and we generate random graphs with a random number of nodes in a given range (13-18) and a random density in the density range. The results shown in table 2 show that for both sparse and dense graphs the `smart` solution outperforms the naive one of about one order of magnitude. A graphical view of how the two solutions perform is shown in the cactus plots in figures [1, 2]

Version	Avg time	edge density
Naive	12.67	[0.1,0.5]
Smart	**1.48**	[0.1, 0.5]
Naive	14.79	[0.5, 0.9]
Smart	**2.06**	[0.5, 0.9]

Tab. 1: Shapley value computation with smart and naive approach

6.2 Nash Equilibria

Just like we did for Shapley Value computation, also in this case we consider two categories of graphs: sparse graphs with edge density in the range [0.1, 0.5] and dense graphs with edge density in the range [0.5, 0.9]. In table 1 we show the results of the comparison for the two types of graphs. It is easy to see that the smart solution provides improvements in the case of sparse graphs while it is equivalent to the naive apporach for dense graphs. The reason is pretty simple and consists in the fact that a dense graph has statistically less nodes that are always cooperating since it is very rare that a node has less that three neighbours in

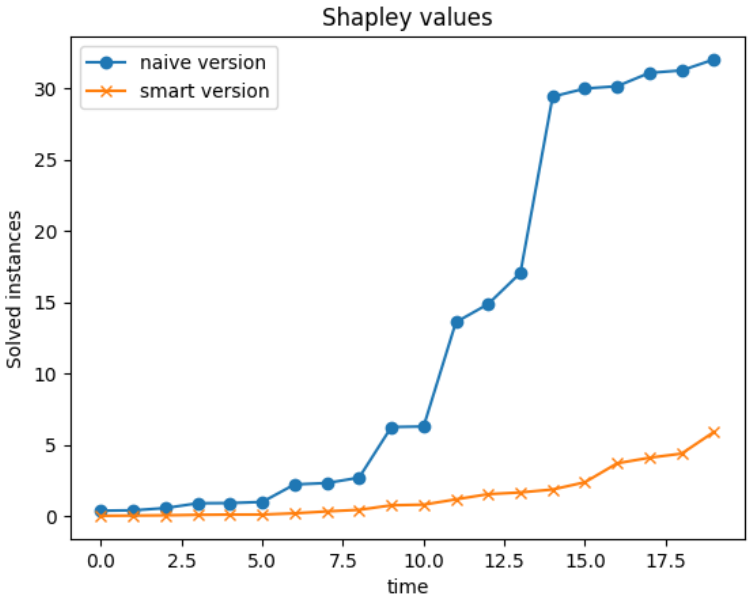

Abb. 1: Shapley value benchmark for sparse graphs

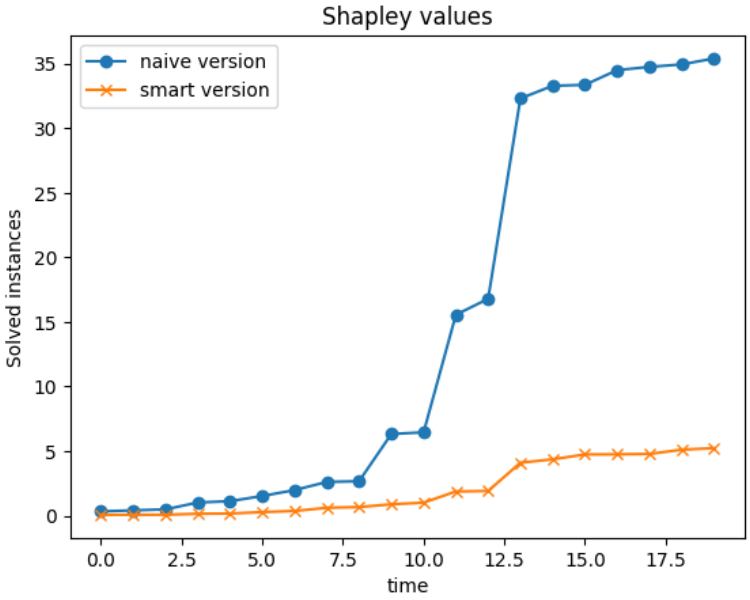

Abb. 2: Shapley value benchmark for dense graphs

a dense graph of say 10 nodes. Therefore for dense graphs the dictionary of strategy profiles is full and there is no player for which the check of best response can be skipped. Plots in figures [3, 4] give a graphical representation of the timings.

Version	Avg time	strat. prof. to check	edge density
Naive	0.60	129400	[0.1,0.5]
Smart	**0.16**	**56900**	[0.1, 0.5]
Naive	0.23	135987	[0.5, 0.9]
Smart	**0.18**	135987	[0.5, 0.9]

Tab. 2: Nash equlibrium computation with smart and naive approach

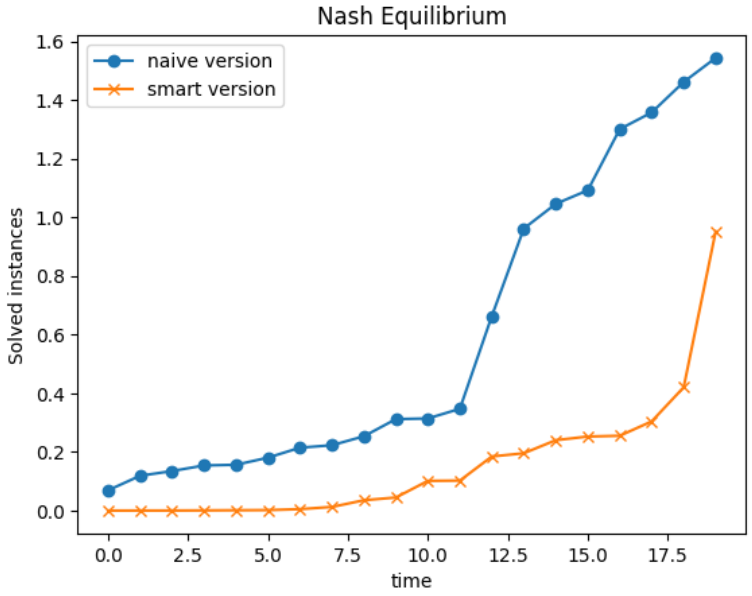

Abb. 3: Nash equilibria benchmark for sparse graphs

6.3 Mechanism Design

The experiment we ran here is very simple. We want to show that the best option a player can do is to declare their real utility. For this ëxperiment"we consider the declaration of players $p_0, ...p_{i-1}, pi + 1, ...p_{n-1}$ fixed and we vary the declaration of the player p_i by increasing it of a fixed quantity for a fixed number of iterations. With this simple trick we compute how the gain of player p_i varies by increasing their declaration. The results shown in Firure 5 consider the player 3 that has a an internal utility of $3 * 10$. It is evident how they gain of p_3 decreases as the declared utility of p_3 increases while the declarations of players $p_0, ...p_{i-1}, pi + 1, ...p_{n-1}$ are kept fixed. However, by giving a first look to the picture we

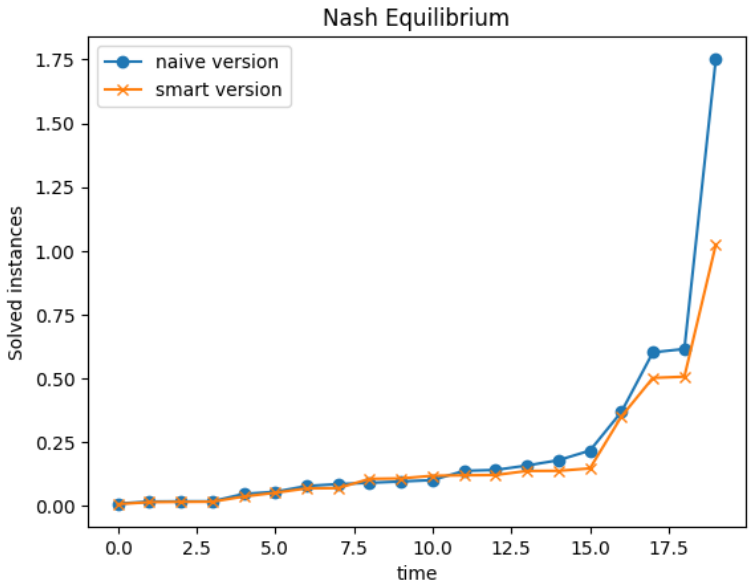

Abb. 4: Nash equilibria benchmark for dense graphs

could experience a little glitch. Again, the real utility of p_3 is 30. Then why does p_3 gain more by declaring a value which is smaller that their real utility? In practice this is not a problem for two reasons. The first is that by declaring a smaller value w.r.t. the real utility a player risks not to be selected in the maximum weighted path which is a bad result. The second reason is that the game is meant to be played just once and not in a sequence of iterations. Therefore no player can optimise their utility over the iterations. This result is pretty much like the one obtained in prisoner dilemma setting [Fo93] in which cooperation emerged as long-run dominant strategy over the classic always defect when the game is played just once.

Literatur

[Fo93] Fogel, D. B.: Evolving Behaviors in the Iterated Prisoner's Dilemma. Evolutionary Computation 1/1, S. 77–97, 1993, ISSN: 1063-6560, eprint: https://direct.mit.edu/evco/article-pdf/1/1/77/1492706/evco.1993.1.1.77.pdf, URL: https://doi.org/10.1162/evco.1993.1.1.77.

[GLS17] Greco, G.; Lupia, F.; Scarcello, F.: The Tractability of the Shapley Value over Bounded Treewidth Matching Games. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. S. 1046–1052, 2017, URL: https://doi.org/10.24963/ijcai.2017/145.

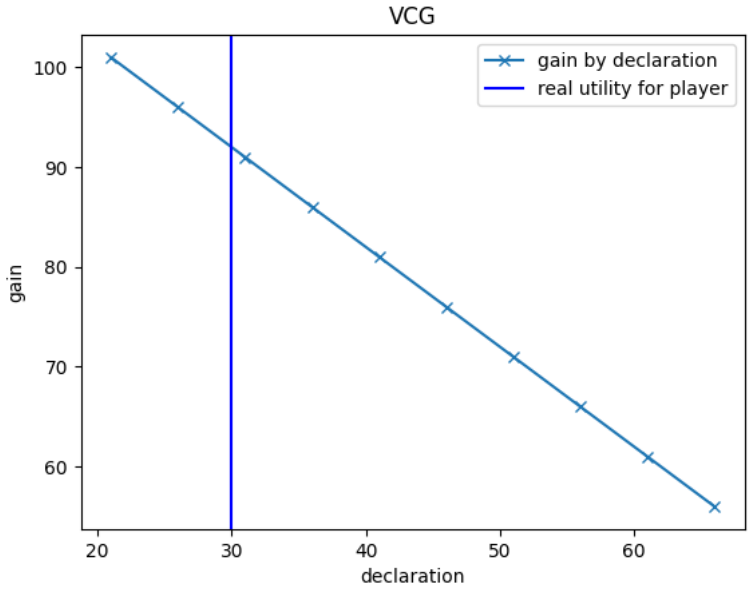

Abb. 5: Gain of player by declaration