Saint Petersburg State University
Department of Mathematical Game Theory
and Statistical Decisions

Kosian David

Master's thesis

Cooperative games on hypergraphs

Specialization 01.04.02 Applied Mathematics and Informatics
Educational Program BM.5644.2017
Master's Program Game Theory and Operations Research

Research advisor:
Doctor of Physics and Mathematics,
Professor L.A.Petrosian

Saint Petersburg
2019

Contents

Introduction

In a classical way for group $N := 1, ..., n$ of agents the economic possibilities of each subgroup are described by cooperative game (N, v), where N is a set of players and v is a characteristic function. The characteristic function shows the power of each coalition. In this paper, we assume the cooperative game with transferable utility or TU-games.

Classically in this game, we assume that each subset of players can decide to cooperate and the total payoff of this cooperation can be distributed among the players. But in many practical situations, not all players can communicate with each other due to some economic, technological or other reasons, thus some coalitions cannot be created. It is the class of TU-games with limited cooperation. The communication structure can be introduced by an undirected graph. In this way, just players who have a link between them can cooperate. These games were first studied in Myerson (1977)[1], he introduced games on a graph and characterized the Shapley value[2]. Hereafter, games with communication structure have received a lot of attention in cooperative game theory. Owen (1986)[3] studied games where the communication structure is a tree. The position value for games where communication structure is given by a graph is introduced by Meessen (1988)[4].

But generally, the communication structure can be given by a graph or hypergraph. For example, it can be some companies or sports teams. Cooperation between two organizations is only possible if they have at least one member in both of them.

The TU-games on hypergraph were studied by Nouweland, Borm and Tijs (1992)[6], they characterized the Mayerson value and the position value for these games. The third value, which is called degree value for the games with hypergraph communication structure was introduced in E.Shan G.Zhang X.Shan (2018)[7]. Many allocation rules for TU-games with a hypergraph communication structure can be proposed based on some different interpretations. The Myerson value highlighting the role of the players, the position value focuses on the role of communication. In this paper, we introduced a new allocation rule for TU-games on the hypergraph.

Chapter 1. Cooperative game without communication structure

In a cooperative game without communication structure, every coalition of players can be made. In the classical cooperative game, we assume that the conditions of the game allow joint actions of player and redistribution of winnings. According to this for all coalitions, we can find the value of coalition, based on some properties. In common the interpretation of this value is the total payoff which coalition can guarantee to itself regardless of the actions of other players who are not in this coalition. So we get the definition of the cooperative TU-game, it is a pair (N, v) where N is a set of players and v is a function that puts the value to each coalition of N.

Characteristic function of game with set of players N we will call a real-valued function $v : 2^N \to R$ and $v(\emptyset) = 0$, defined on all coalitions $S \subseteq N$. Properties of characteristic function:

Monotonicity: $A \subseteq B \Rightarrow v(A) \leq v(B)$. It means that the large coalition get more. Superadditivity: $S \cap T = \emptyset \Rightarrow v(A \cup B) \geq v(A) + v(B)$ for any two coalitions $A \subset N$, $B \subset N$ with no intersection the sum of their values separately is not greater than the value of a union.

The main question in cooperative game is not choose the strategy for each player, the question is how to distribute the total payoff between players. Vector is called imputation $\xi = (\xi_1, \ldots, \xi_n)$ if it satisfies the following conditions where v(j) - is the value of characteristic function for coalition $S = \{j\}$:

$$\xi_j \geq v(j), j \in N,$$

individual rationality, it means that each player gets no less than if he played one without worrying about the actions of other players

$$\sum_{j=1}^{n} \xi_j = v(N),$$

collective rationality, this means that the division exists and the players will not share the nonexistent winnings and will share the total winnings as a whole.

The imputation set for cooperative game (N, v) will denote as $I(N, v)$.

From the definition of imputation, we get that vector $\xi = (\xi_1, \ldots, \xi_n)$ is an imputation then and only then

$$\xi_i = v(i) + \theta_i, i \in N,$$

and

$$\theta_i \geq 0, i \in N, \sum_{i \in N} \theta_i = v(N) - \sum_{i \in n} v(i).$$

If the following statement occurs:

$$\sum_{i \in n} v(i) < v(N)$$

the cooperative game (N, v) is called essential. For nonessential game we have just one imputation $\xi = (v(1), v(2), \ldots, v(n))$

For example the cooperative game with four players where:

$$v(1) = v(2) = 0, v(3) = v(4) = 1,$$

$$v(1, 2) = v(1, 3) = v(1, 4) = v(2, 3) = v(2, 4) = 2$$

$$v(3, 4) = 3, v(1, 2, 3) = v(1, 2, 4) = v(1, 3, 4) = v(2, 3, 4) = 3,$$

$$v(1, 2, 3, 4) = 4$$

is essential because

$$\sum_{i \in n} v(i) = v(1) + v(2) + v(3) + v(4) = 2$$

$$v(N) = 4$$

.

And the cooperative game with four players where:

$$v(1) = v(2) = 1, v(3) = v(4) = 2$$

$$v(1,2) = v(1,3) = v(1,4) = v(2,3) = v(2,4) = 2$$

$$v(3,4) = 3, v(1,2,3) = v(1,2,4) = v(1,3,4) = v(2,3,4) = 4,$$

$$v(1,2,3,4) = 6$$

is nonessential because

$$\sum_{i \in n} v(i) = v(1) + v(2) + v(3) + v(4) = 6$$

$$v(N) = 6$$

.

An imputation ξ dominates an imputation θ in coalition S and we will denote it as $\xi \succ_S \theta$ if

$$\xi_i > \theta_i, i \in S,$$

$$\xi \leq v(S)$$

The first condition means that the imputation ξ is more profitable for all members of coalition S as all members will receive more winnings than by using imputation θ. The second condition means the ability of the coalition to implement this imputation.

An imputation ξ dominates an imputation θ, we will denote as $\xi \succ \theta$ if there are exist coalition $S \subseteq N$ for which is performed $\xi \succ_S \theta$.

On coalition consisting of one player and coalition consisting of all players, dominance is impossible.

There may be a situation where an imputation ξ dominates an imputation θ in coalition $A \subseteq N$, but θ dominates ξ in coalition $B \subseteq N$ and $\xi \neq \theta$

We define the concept of non-dominated imputations. So if the players came to such a imputation of the winning coalition N, that is, the imputation ξ in which no other imputation dominates the imputation ξ. The distribution of winnings of this kind will be stable, in the sense that no coalition A will not be profitable to refuse to cooperate and distribute among themselves the value $v(A)$. This suggests considering as a principle of optimality the set of non-dominated imputations.

The set of non-dominated imputations is called C-core. This set can be empty, if not and the game is essential then this set is infinite. The question of which imputation to choose remains open, but no coalition will have any claims to such an imputation since no matter what imputation we take from the core, no coalition will be able to offer its participants a greater winnings.

For example for the game with three players with the following values of characteristic function

$$v(1) = 6, v(2) = 7.5, v(3) = 9$$

$$v(1,2) = 16, v(1,3) = v(2,3) = 18, v(1,2,3) = 27$$

C-core will be a set of imputations $\xi = (\xi_1, \xi_2, \xi_3)$ which satisfying the conditions

$$\xi_1 \geq 6, \xi_2 \geq 7.5, \xi_3 \geq 9$$

$$\xi_1 \leq 9, \xi_2 \leq 9, \xi_3 \leq 11$$

$$\xi_1 + \xi_2 + \xi_3 = 27$$

This set is non-empty.

The multiplicity of C-core in cooperative games and the strict conditions for the existence of non-dominated imputations, do not solve the problem of choosing the only one imputation in a cooperative game. One of the most well-known cooperative principles of optimality in the game devoid of mentioned disadvantages is the so-called Shapley value.

For the cooperative game (N, v) the vector $\phi(v)$ which defined as follows

$$\phi_i(v) = \sum_{S \subseteq N \setminus i} \frac{|S|!(|N| - |S| - 1)!}{|N|!}(v(S \cup i) - v(S))$$

is called Shapley value.

For example for the game with three players with the following values of characteristic function

$$v(1) = 6, v(2) = 7.5, v(3) = 9$$

$$v(1,2) = 16, v(1,3) = v(2,3) = 18, v(1,2,3) = 27$$

the components of shapley value

$$\phi_1 = \frac{47.5}{6};$$

$$\phi_2 = \frac{52}{6};$$

$$\phi_3 = \frac{62.5}{6}.$$

Therefore, the Shapley value is $\phi(v) = (\frac{47.5}{6}; \frac{52}{6}; \frac{62.5}{6})$ and for this example belongs to the C-core.

Chapter 2. Cooperative game on subclass of hypergraph

2.1 Preliminaries

In this section, we recall some notations and definitions about TU-games and hypergraph. TU-game is a pair (N, v). Characteristic function $v : 2^N R$ and $v(\emptyset) = 0$. We will use $|S|$ to show the cardinality of any $S \in N$.

Hypergraph is a pair (N, H), $H \subseteq \{H \in 2^N \big| |H| \geqslant 2\}$. H is some set of subsets of players N with cardinality more or equal two.

2.2 Definition of the game

Let $N = \{1, \ldots, n-1, c\}$ be a player set. Communication possibilities described by hypergraph (N, H) In this part we will consider a special communication structure which given by

$$\overline{H} \subseteq \{H \in 2^N \big| |H| \geqslant 2,\ H_j \cap H_k = c;\ j \neq k\ \forall H_j, H_k \in H\}.$$

The interpretation of this structure is there is just one player who included in all hyperlinks and other players included just in one of them. The communication is only possible between the players in hyperlinks. It can be also interpreted as the central player has some companies with workers.An example of this hypergraph shown in fig.1

Denote the numbers of hyperlink in communication structure by L. Also denote the central-player by c. Let Γ_i be a set of players which included in hyperlink H_i except player c and U_i — the set of their strategies. Also denote a strategy of simple-player j as u^j. A strategy of player c from his set of strategies we will denote by $u_c \in \mathfrak{U}_c$. We define the payoff function of simple-player j in hyperlink H_i in this way

$$h_j(U_i, u_c) = K_j(U_i, u_c)$$

where K_j — payoff of player j which is defined on hyperlink which include

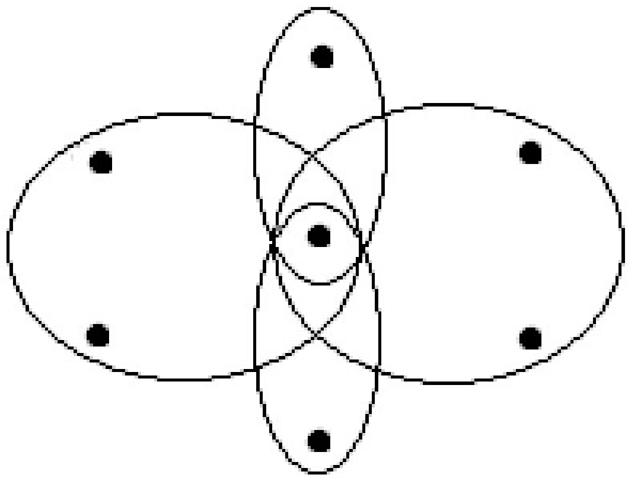


Figure 1: An example of this hypergraph.

player j. The payoff function of central-player c:

$$h_c(U_1, U_2, \dots, U_L, u_c) = K_c^1(U_1, u_c) + K_c^2(U_2, u_c) + \dots + K_c^L(U_L, u_c)$$

2.3 Cooperation

Now consider the case when the players agree to cooperate. It means that they will choose their strategies to maximize the sum of their payoffs

$$\sum_{k \in N} h_k = \sum_{i=1}^{L} \sum_{m \in H_i} K_m \left(U_i, u_c\right) + \sum_{j=1}^{L} K_c^j(U_j, u_c).$$

We suppose transferable payoffs. Thus the main question is how to allocate the total payoff between players. We will do it in three steps. On the first one, we construct a new cooperative game where we suppose hyperlinks as players. We will create a characteristic function for all coalitions in this game. After that we solve this game with some allocation rule, in this paper we used a solution with equal excess. So we get the payoff for all hyperlinks. The second step is to allocate this payoff between the members in a hyperlink. To solve this problem we will use the proportional solution. The last step is to find the total payoff for the central player. It will be the sum of his payoffs from all hyperlinks.

2.3.1 First step

For now we consider the game where the players are hyperlinks from the given communication structure.The set of hyperlinks which are the players in this part we will denote as $\mathcal{H}$. $\mathcal{S} \subseteq \mathcal{H}$ is coalition from this set of links. We will define the characteristic function which defined for all coalitions of this players as follows:

$$V(\mathcal{S}) = \sum_{i:\, H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K_j(\widetilde{U}_i, \hat{u}_c) + \sum_{i:\, H_i \in \mathcal{S}} K_c^i(\widetilde{U}_i, \hat{u}_c),$$

where $\hat{u}_c$ the solution of this maximization problem:

$$\max_{u_c} \max_{U_i} \left(\sum_{i:\, H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i} K_j(U_i, u_c) + \sum_{i:\, H_i \notin \mathcal{S}} K_c^i(U_i, u_c) \right) =$$

$$= \sum_{i:\, H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i} K_j(\widehat{U}_i, \widehat{u}_c) + \sum_{i:\, H_i \notin \mathcal{S}} K_c^i(\widehat{U}_i, \widehat{u}_c),$$

and $\widetilde{U}_i$ the solution of:

$$\max_{U_i} \left(\sum_{i:\, H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K_j(U_i, \widehat{u}_c) + \sum_{i:\, H_i \in \mathcal{S}} K_c^i(U_i, \widehat{u}_c) \right) =$$

$$= \sum_{i:\, H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K_j(\widetilde{U}_i, \widehat{u}_c) + \sum_{i:\, H_i \in \mathcal{S}} K_c^i(\widetilde{U}_i, \widehat{u}_c)$$

$$V(\mathcal{H}) = \max_{u_c} \max_{U_i} \left(\sum_{i:\, H_i \in \mathcal{H}} \sum_{j \in \Gamma_i} K_j(U_i, u_c) + \sum_{i:\, H_i \in \mathcal{H}} K_c^i(U_i, u_c) \right)$$

This characteristic function can be interpreted as follows. The central player wants to maximize the total payoff of all players in hyperlinks which are not in S. Based on this, the central player's $\widehat{u}_c$ strategy is chosen, assuming that in the worst case the central player will play this strategy, players from S seek to maximize their total payoff. Thus, we have determined the characteristic function for all coalitions of the hyperlinks. The next step is to find the winnings for each hyperlink. For this, we will use a solution with equal excess.

$$\xi_{H_j} = V(H_J) + \frac{V(\mathcal{H}) - \sum\limits_{i \in \mathcal{H}} V(H_i)}{L}, \quad j = \overline{1, L}.$$

2.3.2 Second step

After the previous step the payoffs for each hyperlink have been received, the next step will be the distribution of this payoff between the players in each hyperlink. At this step, we get L cooperative games, for each we uniformly define the characteristic function for coalitions of players and the optimality principle. The characteristic function in these games will be determined in accordance with the approach described in [9]. The idea is quite simple: the characteristic function in this case shows the maximum gain that a coalition can receive, provided that all other players play against it. As an optimality principle, we take a proportional solution. Consider a game on the j hyperlink. We denote the set of players on this hyperlink, including the central one, by N_j. We assume that $v^j(N_j) = \xi_{H_j}$. Let us define the value of the characteristic function for each of the simple players on this j hyperlink as

$$v^j(i) = \max_{u^i} \min_{u_c \bigcup U_j \setminus u^i} K_i(U_j, u_c).$$

for central-player on the same hyperlink

$$v^j(c) = \max_{u_c} \min_{U_j} K_c^j(U_j, u_c).$$

Now we can define the payoff of each simple-player on the hyperlink j as:

$$\mathcal{E}_i^j = \frac{v^j(i)}{\sum\limits_{k \in N_j} v^j(k)} v(N_j) = \frac{v^j(i)}{\sum\limits_{k \in N_j} v^j(k)} \xi_{H_i}.$$

The payoff of the central-player on the hyperlink j we will define by

$$\mathcal{E}_c^j = \frac{v^j(c)}{\sum\limits_{k \in N_j} v^j(k)} v(N_j) = \frac{v^j(c)}{\sum\limits_{k \in N_j} v^j(k)} \xi_{H_i}.$$

2.3.3 Third step

Thus we already defined the payoffs of all simple-players. The payoff of the central-player we will find as a sum of his payoffs on each hyperlink.

$$\mathcal{E}_c = \sum_{j=1}^{L} \mathcal{E}_c^j.$$

This is the main idea of this work.

2.4 Example

For better understanding we will use this solution on the example. Consider the cooperative game with player set $N = \{1, 2, 3, 4, c\}$ and hypergraph $H_1 = \{1, 2, c\}, H_2 = \{3, 4, c\}$ which is shown on fig.2 .

For this example we consider that in each hyperlink players have bimatrix game between each other. It means that for simple-player j in hyperlink H_i payoff function is

$$h_j(U_i, u_c) = K_j(U_i, u_c) = \sum_{k: u^k \in U_i \setminus u^j} K_j(u^k, u^j) + K_j(u^j, u_c),$$

and for central player the payoff function

$$h_c(U_1, U_2, \dots, U_L, u_c) = K_c^1(U_1, u_c) + K_c^2(U_2, u_c) + \dots + K_c^L(U_L, u_c)$$

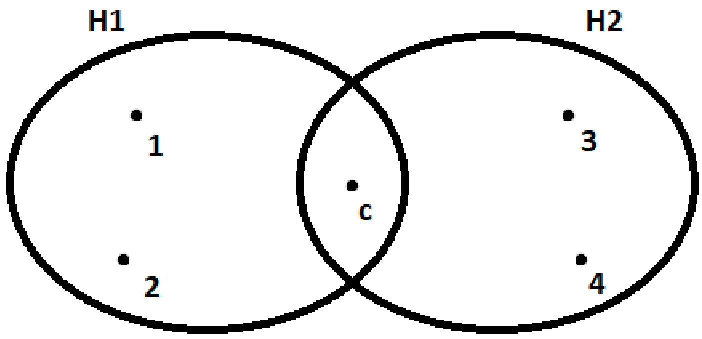


Figure 2: Communication structure

where

$$K_c^i(U_1, u_c) = \sum_{k:u^k \in U_i} K_c^i(u^k, u_c)$$

Lets define the bimatrix game for each pair of linked players. We will write a bimatrix 2×2 for player i and j where i chooses the row and j chooses column. We consider that all players have the set of strategies (A, B).

For players 1 and c

$$\begin{pmatrix} 4\backslash 8 & 3\backslash 6 \\ 1\backslash 3 & 5\backslash 6 \end{pmatrix}$$

For players 2 and c

$$\begin{pmatrix} 3\backslash 6 & 5\backslash 5 \\ 0\backslash 2 & 4\backslash 8 \end{pmatrix}$$

For players 1 and 2

$$\begin{pmatrix} 6\backslash 8 & 6\backslash 0 \\ 4\backslash 3 & 0\backslash 6 \end{pmatrix}$$

For players 3 and c

$$\begin{pmatrix} 8\backslash 0 & 6\backslash 10 \\ 3\backslash 6 & 9\backslash 3 \end{pmatrix}$$

For players 4 and c

$$\begin{pmatrix} 5\backslash 2 & 8\backslash 9 \\ 7\backslash 2 & 6\backslash 5 \end{pmatrix}$$

For players 3 and 4

$$\begin{pmatrix} 0\backslash 1 & 10\backslash 4 \\ 7\backslash 0 & 3\backslash 8 \end{pmatrix}$$

First step. Firstly we will find the value of characteristic function for all coalitions of hyperlinks.

$$V(H_1) = \sum_{j\in\Gamma_1} K_j(\widetilde{U}_1, \hat{u}_c) + K_c^1(\widetilde{U}_1, \hat{u}_c),$$

where $\hat{u}_c$ the solution of this maximization problem:

$$\max_{u_c}\max_{U_2}\left(\sum_{j\in\Gamma_2} K_j(U_2, u_c) + K_c^2(U_2, u_c)\right) =$$

$$= \sum_{j\in\Gamma_2} K_j(\widehat{U}_2, \widehat{u}_c) + K_c^2(\widehat{U}_2, \widehat{u}_c) = 41$$

In this example $\hat{u}_c = B$ next we find $\widetilde{U}_1$ it is the solution of:

$$\max_{U_1}\left(\sum_{j\in\Gamma_i} K_j(U_1, B) + K_c^1(U_1, B)\right) = 33$$

Thus $V(H_1) = 33$

$$V(H_2) = \sum_{j\in\Gamma_2} K_j(\widetilde{U}_2, \hat{u}_c) + K_c^2(\widetilde{U}_2, \hat{u}_c),$$

where $\hat{u}_c$ the solution of this maximization problem:

$$\max_{u_c}\max_{U_1}\left(\sum_{j\in\Gamma_1} K_j(U_1, u_c) + K_c^1(U_1, u_c)\right) =$$

$$= \sum_{j \in \Gamma_1} K_j(\widehat{U}_1, \widehat{u}_c) + K_c^1(\widehat{U}_1, \widehat{u}_c) = 35$$

In this example $\hat{u}_c = A$ next we find $\widetilde{U}_2$ it is the solution of:

$$\max_{U_2} \left(\sum_{j \in \Gamma_2} K_j(U_2, A) + K_c^1(U_2, A) \right) = 31$$

Thus $V(H_2) = 31$

$$V(\mathcal{H}) = \max_{u_c} \max_{U_i} \left(\sum_{i:\, H_i \in \mathcal{H}} \sum_{j \in \Gamma_i} K_j(U_i, u_c) + \sum_{i:\, H_i \in \mathcal{H}} K_c^i(U_i, u_c) \right) = 74$$

Now we use the solution with equal excess to get winnings for hyperlinks

$$\xi_{H_j} = V(H_i) + \frac{V(\mathcal{H}) - \sum_{i \in \mathcal{H}} V(H_i)}{L}, \quad j = \overline{1, L}.$$

$$\xi_{H_1} = V(H_1) + \frac{V(\mathcal{H}) - (V(H_1) + V(H_2))}{2} = 38$$

$$\xi_{H_2} = V(H_2) + \frac{V(\mathcal{H}) - (V(H_1) + V(H_2))}{2} = 36$$

Second step. Now we need to solve two cooperative game as an optimality principle we will use proportional solution. For the game on hyperlink H_1 a characteristic function for players 1,2 and c

$$v^1(1) = \max_{u^1} \min_{u_c \bigcup U_1 \setminus u^1} K_i(U_1, u_c) = \max_{u^1} \min_{u_c, u^2} (K_1(u^1, u_c) + K_1(u^1, u^2)) = 10.$$

$$v^1(2) = \max_{u^2} \min_{u_c \bigcup U_1 \setminus u^2} K_i(U_1, u_c) = \max_{u^2} \min_{u_c, u^1} (K_2(u^2, u_c) + K_2(u^1, u^2)) = 6.$$

$$v^1(c) = \max_{u_c} \min_{U_1} K_c^1(U_1, u_c) = \max_{u_c} \min_{u^1, u^2} (K_c^1(u^2, u_c) + K_c^1(u^1, u_c)) = 11$$

$$v^1(N_1) = \xi_{H_1} = 38$$

$$\mathcal{E}_1^1 = \frac{v^1(1)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{380}{27}$$

$$\mathcal{E}_1^2 = \frac{v^1(2)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{228}{27}$$

$$\mathcal{E}_1^c = \frac{v^1(c)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{418}{27}$$

For the game on hyperlink H_2 a characteristic function for players 3,4 and c

$$v^2(3) = \max_{u^3} \min_{u_c \bigcup U_2 \setminus u^3} K_i(U_2, u_c) = \max_{u^3} \min_{u_c, u^4} (K_3(u^3, u_c) + K_3(u^3, u^4)) = 15.$$

$$v^2(4) = \max_{u^4} \min_{u_c \bigcup U_2 \setminus u^4} K_i(U_2, u_c) = \max_{u^2} \min_{u_c, u^3} (K_4(u^4, u_c) + K_4(u^3, u^4)) = 11.$$

$$v^2(c) = \max_{u_c} \min_{U_2} K_c^2(U_2, u_c) = \max_{u_c} \min_{u^1, u^2} (K_c^2(u^3, u_c) + K_i(u^4, u_c)) = 8$$

$$v^2(N_2) = \xi_{H_2} = 36$$

$$\mathcal{E}_2^3 = \frac{v^2(3)}{v^2(3) + v^2(4) + v^2(c)} v^2(N_2) = \frac{540}{34}$$

$$\mathcal{E}_2^4 = \frac{v^2(4)}{v^1(3) + v^2(4) + v^2(c)} v^2(N_2) = \frac{396}{34}$$

$$\mathcal{E}_2^c = \frac{v^2(c)}{v^1(3) + v^2(4) + v^2(c)} v^2(N_2) = \frac{288}{34}$$

Third step. Now we need to sum the payoffs of central player from each hyperlink

$$\mathcal{E}_c = \sum_{j=1}^{L} \mathcal{E}_c^j = \mathcal{E}_1^c + \mathcal{E}_2^c = \frac{288}{34} + \frac{418}{27}$$

Chapter 3. Generalization of the game

3.1 Preliminaries

We already construct the TU-game where communication structure is given by hypergraph with special properties. In this part we will generalize this game. Now we need to refresh some information about hypergraph.

The reduction of hypergraph (N, H) is called hypergraph $(N, H^{'})$ which is obtained from the original by removing all hyperlinks that are completely contained in other hyperlinks. Hypergraph is called reduced if it is equivalent to its reduction, that is, it does not have a hyperlink inside other hyperlinks.

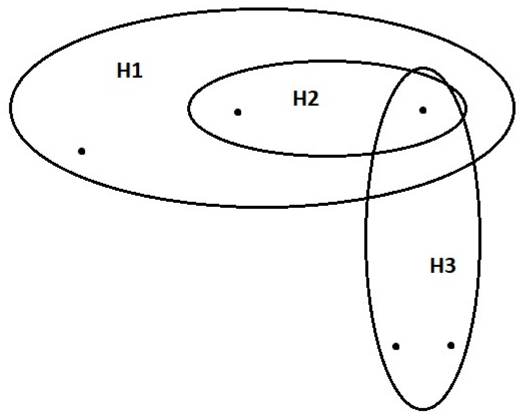


Figure 3: An example of not reduced hypergraph.

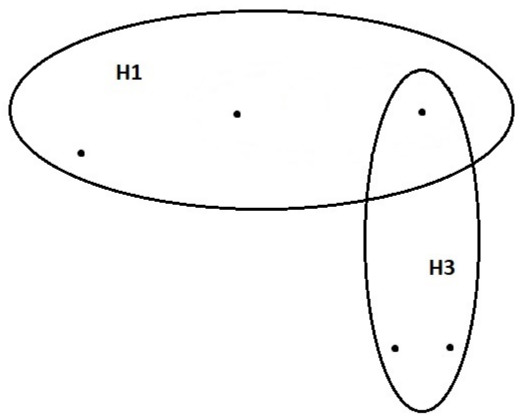


Figure 4: Reduction of hypergraph on figure 3.

A simple cycle with length s in hypergraph (N, H) is a sequence

$$(H_0, n_0, H_1, \ldots, H_{s1}, n_{s1}, H_s),$$

where $H_0, \ldots, H_{s1}$ different hyperlinks, hyperlink H_s coincides with H_0, $n_0, \ldots, n_{s-1}$ different vertexes, and $n_i \in H_i \cap H_{i+1}$ for all $i = 0, \ldots, s-1$.

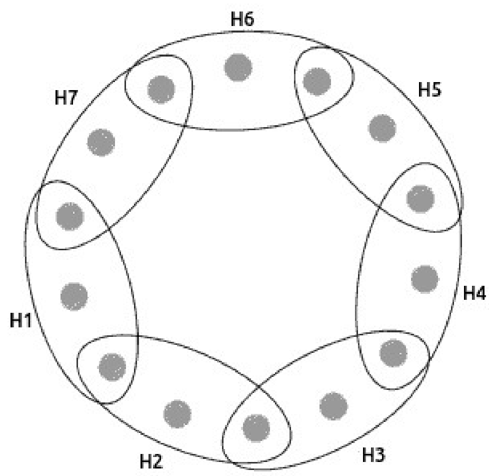


Figure 5: An a simple cycle on hypergraph.

A first definition of acyclicity for hypergraphs was given by Claude Berge[11] a hypergraph is acyclic if its incidence graph is acyclic.

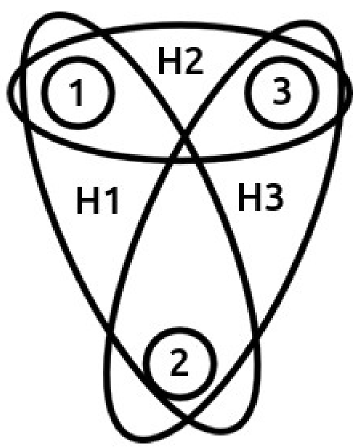


Figure 6: A hypergraph with a cycle.

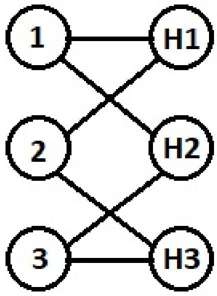


Figure 7: Incidence graph of hypergraph on fig.6.

3.2 Definition of the game

In this part, we will construct the game where communication structure defined by acyclic reduced hypergraph (N, H). An interpretation of this communication structure can be that there are some managers who work with some companies and each company has workers who work just on them. An example of this communication is given on pic.2.

Let $N := \{1, \ldots, n-m, c_1, \ldots, c_m\}$ be a set of players. Denote the numbers of hyperlinks in communication structure by L as before. The players which included just in one hyperlink we will call simple-players, other will be called complex-players. To construct the game we need to enter some new notations.

Let $\mathfrak{u}^i$ is strategy of simple-player i from the set of his strategies $\mathfrak{U}^i$. Also denote as $\mathfrak{u}^{c_j}$ a strategy of complex-player j from the set of his strategies $\mathfrak{U}^{c_j}$. The set of simple-players strategies in hyperlink H_i we will denote as U_i and the set of complex-players strategies in this hyperlink as U_i^c. The payoff function for simple-player j in hyperlink H_i denote as $K^j(U_i, U_i^c)$, and the payoff function for complex-player j in hyperlink H_i denote by $K_i^{c_j}(U_i, U_i^c)$. Now we can define the total payoff function of each player. For all simple-players for example j which included in hyperlink the total payoff will be

$$h^j = K^j(U_i, U_i^c)$$

the total payoff function of complex-player j we define as

$$h^{c_j} = \sum_{i:c_j \in H_i} K_i^{c_j}(U_i, U_i^c)$$

3.3 Cooperation

As before we consider a cooperative game where the players agree to choose their strategies together to maximize the total sum of theirs payoffs. The total sum is equal:

$$\sum_{i=1}^{n-m} h^i + \sum_{i=1}^{m} h^{c_i} = \sum_{i=1}^{L} \sum_{j \in H_i} K^j(U_i, U_i^c) + \sum_{i=1}^{L} \sum_{j:c_j \in H_i} K_i^{c_j}(U_i, U_i^c)$$

Now we will do the same as before. Firstly we consider the cooperation game where players are hyperlinks. We will define a characteristic function for any coalition of hyperlinks, after that we will use an allocation rule and get payoff for each hyperlink. Next step is consider L cooperation games and get payoff for each player. Finally we will find a total payoffs for any complex-player as a sum from his payoffs from each hyperlink in which he exist.

3.3.1 First step

We consider the cooperative game with hyperlinks as players. To define the characteristic function for all coalitions we need to denote some new notations. Let Γ_i be a set of simple-players in hyperlink H_i, Γ_i^c is a set of complex-players in hyperlink H_i. Also denote a set of hyperlinks which include complex-player j as B_{c_j}. For any coalition S we will make a partition on each hyperlink in s of complex-players set in this hyperlink. The set of strategies in hyperlink H_i of complex-players who have all the edges in which they are included in the coalition we denote as U_i^{cf} others by U_i^{cn}, $U_i^{cf} {}_i^{cn} = U_i^c$. The set of hyperlinks which are the players in this part we will denote as $\mathcal{H}$. $\mathcal{S} \subseteq \mathcal{H}$ is coalition from this set of links. Now we can define the characteristic function for all coalitions of hyperlinks as follows

$$V(\mathcal{S}) = \sum_{i:H_i\in\mathcal{S}}\sum_{j\in\Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i\in\mathcal{S}}\sum_{j\in\Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n})$$

where $\hat{U}_i^{c^n}$ the solution of this maximization problem:

$$\max_{U_i^{c^n}}\max_{U_i}\left(\sum_{i:H_i\notin\mathcal{S}}\sum_{j\in\Gamma_i} K^j(U_i, U_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}}\sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^n})\right) =$$

$$= \sum_{i:H_i\notin\mathcal{S}}\sum_{j\in\Gamma_i} K^j(\widehat{U}_i, \widehat{U}_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}}\sum_{j\in\Gamma_i^c} K_i^{c_j}(\widehat{U}_i, \widehat{U}_i^{c^n})$$

and $\widetilde{U}_i$ and $\widetilde{U}_i^{c^f}$ the solution of:

$$\max_{U_i^{c^f}}\max_{U_i}\left(\sum_{i:H_i\in\mathcal{S}}\sum_{j\in\Gamma_i} K^j(U_i, U_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i\in\mathcal{S}}\sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^f}, \widehat{U}_i^{c^n})\right) =$$

$$= \sum_{i:H_i\in\mathcal{S}}\sum_{j\in\Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i\in\mathcal{S}}\sum_{j\in\Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n})$$

for the grand coalition we have:

$$V(\mathcal{H}) = \max_{U_i^c}\max_{U_i}\left(\sum_{i:H_i\in\mathcal{H}}\sum_{j\in\Gamma_i} K^j(U_i, U_i^c,) + \sum_{i:H_i\in\mathcal{H}}\sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^c)\right)$$

As allocation rule here we will use Shapley value. Notice that in this step we can use any allocation rule from classic cooperative theory.

$$\phi_{H_i}(V) = \sum_{\mathcal{S}\subseteq\mathcal{H}\setminus H_i} \frac{|\mathcal{S}|!(|\mathcal{N}| - |\mathcal{S}| - 1)!}{\mathcal{N}!}(V(\mathcal{S}\cup H_i) - V(\mathcal{S}))$$

3.3.2 Second step

From the previous step we get winnings for each hyperlink in our hypergraph. Now we consider a cooperation game on each hyperlink with players which included in it. It means that now we have L independent cooperative games. As an optimality principle we will use proportional solution. We denote the set of players on this hyperlink by N_j. We assume that $v^j(N_j) = \phi_{H_j}$. Let is define the value of the characteristic function for each of the simple players on this j hyperlink as

$$v^j(i) = \max_{u^i} \min_{U_j^c \bigcup U_j \setminus u^i} K^i(U_j, U_j^c).$$

for complex-player i on the same hyperlink

$$v^j(c_i) = \max_{u^{c_i}} \min_{U_j^c \setminus u^{c_i} \bigcup U_j} K_j^{c_i}(U_j, U_j^c).$$

Now we can define the payoff of each simple-player on the hyperlink j as:

$$\mathcal{E}_i^j = \frac{v^j(i)}{\sum\limits_{k \in N_j} v^j(k)} v^j(N_j) = \frac{v^j(i)}{\sum\limits_{k \in N_j} v^j(k)} \phi_{H_i}.$$

The payoff of complex-player i on the hyperlink j we will define by

$$\mathcal{E}_{c_i}^j = \frac{v^j(c_i)}{\sum\limits_{k \in N_j} v^j(k)} v^j(N_j) = \frac{v^j(c_i)}{\sum\limits_{k \in N_j} v^j(k)} \phi_{H_i}.$$

3.3.3 Third step

Now we get total payoffs for each simple-player. The total payoff for each complex-player is the sum of his payoffs from each hyperlink in which it is included.

$$\mathcal{E}_{c_i} = \sum_{j: H_j \in B_{c_i}} \mathcal{E}_{c_i}^j$$

3.4 Example

Consider the cooperative game with player set $N = \{1, 2, 3, 4, c_1, c_2\}$ and hypergraph $H_1 = \{1, 2, c_1\}, H_2 = \{3, c_1, c_2\}, H_3 = \{c_2, 4\}$ which is shown on fig.8.

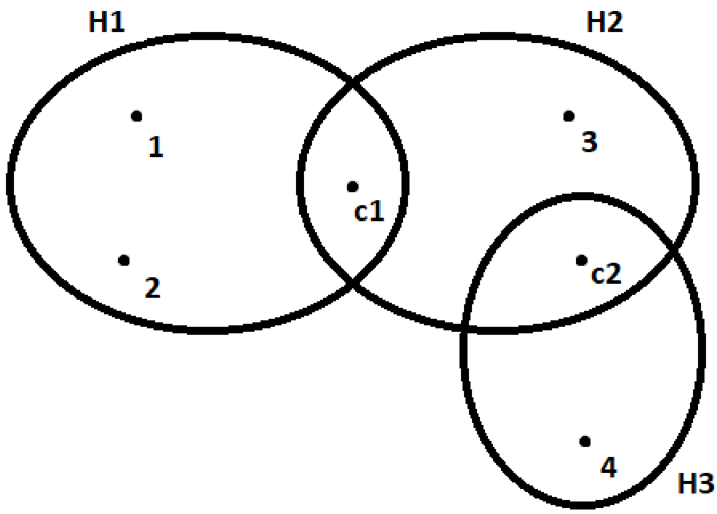


Figure 8: Communication structure

For this example we consider that in each hyperlink players have bimatrix game between each other. It means that for simple-player i in hyperlink H_j payoff function is

$$h^i(U_j, U_j^c) = K^i(U_j, U_j^c) = \sum_{k:u^k \in U_j \setminus u^i} K^i(u^k, u^i) + \sum_{k:u^{c_k} \in U_j^c} K^i(u^i, u^{c_k}),$$

and for central player the payoff function

$$h^{c_j} = \sum_{i:c_j \in H_i} K_i^{c_j}(U_i, U_i^c)$$

where

$$K_j^{c_i}(U_j, U_j^c) = \sum_{k:u^k \in U_j} K^i(u^k, u^{c_i}) + \sum_{k:u^{c_k} \in U_j^c \setminus u^{c_i}} K^i(u^{c_i}, u^{c_k})$$

Let's define the bimatrix game for each pair of linked players. We will write a bimatrix 2×2 for player i and j where i chooses the row and j chooses column. We consider that all players have a set of strategies (A, B).

For players 1 and c_1

$$\begin{pmatrix} 4\backslash 8 & 3\backslash 6 \\ 1\backslash 3 & 5\backslash 6 \end{pmatrix}$$

For players 2 and c_1

$$\begin{pmatrix} 3\backslash 6 & 5\backslash 5 \\ 0\backslash 2 & 4\backslash 8 \end{pmatrix}$$

For players 1 and 2

$$\begin{pmatrix} 6\backslash 8 & 6\backslash 0 \\ 4\backslash 3 & 0\backslash 6 \end{pmatrix}$$

For players 3 and c_1

$$\begin{pmatrix} 8\backslash 0 & 6\backslash 10 \\ 3\backslash 6 & 9\backslash 3 \end{pmatrix}$$

For players c_2 and c_1

$$\begin{pmatrix} 5\backslash 2 & 8\backslash 9 \\ 7\backslash 2 & 6\backslash 5 \end{pmatrix}$$

For players 3 and c_2

$$\begin{pmatrix} 0\backslash 1 & 10\backslash 4 \\ 7\backslash 0 & 3\backslash 8 \end{pmatrix}$$

For players 4 and c_2

$$\begin{pmatrix} 1\backslash 4 & 2\backslash 7 \\ 4\backslash 0 & 3\backslash 5 \end{pmatrix}$$

First step. Firstly we will find the value of the characteristic function for all coalitions of hyperlinks. In coalition $\mathcal{S} = \{H_1\}$, $U_1^{c^n} = U_1^c = (u_1^c)$ then the value of characteristic function of this coalition is equal

$$V(H_1) = \sum_{j\in\Gamma_1} K^j(\widetilde{U_1}, \widehat{U}_1^{c^n}) + \sum_{j\in\Gamma_1^c} K_1^{c_j}(\widetilde{U}_1, \widehat{U}_1^{c^n})$$

$$\max_{U_i^{c^n}} \max_{U_i} \left(\sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i} K^j(U_i, U_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^n}) \right) =$$

$$= \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i} K^j(\widehat{U}_i, \widehat{U}_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(\widehat{U}_i, \widehat{U}_i^{c^n}) = 50$$

from this we get $\widehat{U}_1^{c^n} = (\widehat{u}^{c_1}) = B$

$$\max_{U_1} \left(\sum_{j\in\Gamma_1} K^j(U_1, \widehat{U}_1^{c^n}) + \sum_{j\in\Gamma_1^c} K_1^{c_j}(U_1, \widehat{U}_1^{c^n}) \right) =$$

$$= \sum_{j\in\Gamma_1} K^j(\widetilde{U_1}, \widehat{U}_1^{c^n}) + \sum_{j\in\Gamma_1^c} K_1^{c_j}(\widetilde{U}_1, \widehat{U}_1^{c^n}) = 33$$

Thus we get $V(H_1) = 33$.

In coalition $\mathcal{S} = \{H_2\}$, $U_2^{c^n} = U_2^c = (u^{c_1}, u^{c_2})$ then the value of characteristic function of this coalition is equal

$$V(H_2) = \sum_{j\in\Gamma_2} K^j(\widetilde{U_2}, \widehat{U}_2^{c^n}) + \sum_{j\in\Gamma_2^c} K_2^{c_j}(\widetilde{U}_2, \widehat{U}_2^{c^n})$$

$$\max_{U_i^{c^n}} \max_{U_i} \left(\sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i} K^j(U_i, U_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^n}) \right) =$$

$$= \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i} K^j(\widehat{U}_i, \widehat{U}_i^{c^n}) + \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(\widehat{U}_i, \widehat{U}_i^{c^n}) = 44$$

from this we get $\widehat{U}_2^{c^n} = (\widehat{u}^{c_1}, \widehat{u}^{c_2}) = (A, B)$

$$\max_{U_2} \left(\sum_{j \in \Gamma_2} K^j(U_2, \widehat{U}_2^{c^n}) + \sum_{j \in \Gamma_2^c} K_2^{c_j}(U_2, \widehat{U}_2^{c^n}) \right) =$$

$$= \sum_{j \in \Gamma_2} K^j(\widetilde{U_2}, \widehat{U}_2^{c^n}) + \sum_{j \in \Gamma_2^c} K_i^{c_j}(\widetilde{U}_2, \widehat{U}_2^{c^n}) = 31$$

Thus we get $V(H_2) = 31$.

In coalition $\mathcal{S} = \{H_3\}$, $U_3^{c^n} = U_3^c = (u^{c_2})$ then the value of characteristic function of this coalition is equal

$$V(H_3) = \sum_{j \in \Gamma_3} K^j(\widetilde{U_3}, \widehat{U}_3^{c^n}) + \sum_{j \in \Gamma_3^c} K_3^{c_j}(\widetilde{U}_3, \widehat{U}_3^{c^n})$$

$$\max_{U_i^{c^n}} \max_{U_i} \left(\sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i} K^j(U_i, U_i^{c^n}) + \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^n}) \right) =$$

$$= \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i} K^j(\widehat{U}_i, \widehat{U}_i^{c^n}) + \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(\widehat{U}_i, \widehat{U}_i^{c^n}) = 74$$

from this we get $\widehat{U}_3^{c^n} = (\widehat{u}^{c_2}) = (B)$

$$\max_{U_3} \left(\sum_{j \in \Gamma_3} K^j(U_3, \widehat{U}_3^{c^n}) + \sum_{j \in \Gamma_3^c} K_3^{c_j}(U_3, \widehat{U}_3^{c^n}) \right) =$$

$$= \sum_{j \in \Gamma_3} K^j(\widetilde{U_3}, \widehat{U}_3^{c^n}) + \sum_{j \in \Gamma_3^c} K_i^{c_j}(\widetilde{U}_3, \widehat{U}_3^{c^n}) = 9$$

Thus we get $V(H_3) = 9$.

In coalition $\mathcal{S} = \{H_1, H_2\}$, $U_1^{c^f} = U_1^c = (u^{c_1}), U_2^{c^n} = (u^{c_2}), U_2^{c^f} = (u^{c_1})$ then the value of characteristic function of this coalition is equal

$$V(\mathcal{S}) = V(\{H_1, H_2\}) = \sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n})$$

$$\max_{U_i^{c^n}} \max_{U_i} \left(\sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i} K^j(U_i, U_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^n}) \right) =$$

$$= \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i} K^j(\widehat{U}_i, \widehat{U}_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(\widehat{U}_i, \widehat{U}_i^{c^n}) =$$

From this we get $\widehat{U}_2^{c^n} = (u^{c_2}) = B$

$$\max_{U_i^{c^f}} \max_{U_i} \left(\sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i} K^j(U_i, U_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^f}, \widehat{U}_i^{c^n}) \right) =$$

$$= \sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) = 74$$

Thus we get $V(\{H_1, H_2\}) = 74$.

In coalition $\mathcal{S} = \{H_2, H_3\}$, $U_3^{c^f} = U_3^c = (u^{c_3}), U_2^{c^n} = (u^{c_1}), U_2^{c^f} = (u^{c_2})$ then the value of characteristic function of this coalition is equal

$$V(\mathcal{S}) = V(\{H_2, H_3\}) = \sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i\in\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n})$$

$$\max_{U_i^{c^n}} \max_{U_i} \left(\sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i} K^j(U_i, U_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^n}) \right) =$$

$$= \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i} K^j(\widehat{U}_i, \widehat{U}_i^{c^n}) + \sum_{i:H_i\notin\mathcal{S}} \sum_{j\in\Gamma_i^c} K_i^{c_j}(\widehat{U}_i, \widehat{U}_i^{c^n}) =$$

From this we get $\widehat{U}_2^{c^n} = (u^{c_1}) = A$

$$\max_{U_i^{c^f}} \max_{U_i} \left(\sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K^j(U_i, U_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^f}, \widehat{U}_i^{c^n}) \right) =$$

$$= \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) = 40$$

Thus we get $V(\{H_2, H_3\}) = 40$.

In coalition $\mathcal{S} = \{H_1, H_3\}$, $U_1^{c^n} = U_1^c = (u^{c_1}), U_3^{c^n} = U_3^c = (u^{c_2})$ then the value of characteristic function of this coalition is equal

$$V(\mathcal{S}) = V(\{H_1, H_3\}) = \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n})$$

$$\max_{U_i^{c^n}} \max_{U_i} \left(\sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i} K^j(U_i, U_i^{c^n}) + \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^n}) \right) =$$

$$= \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i} K^j(\widehat{U}_i, \widehat{U}_i^{c^n}) + \sum_{i:H_i \notin \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(\widehat{U}_i, \widehat{U}_i^{c^n}) =$$

From this we get $\widehat{U}_3^{c^n} = (u^{c_2}) = B$, and $\widehat{U}_1^{c^n} = (u^{c_1}) = B$

$$\max_{U_i^{c^f}} \max_{U_i} \left(\sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K^j(U_i, U_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(U_i, U_i^{c^f}, \widehat{U}_i^{c^n}) \right) =$$

$$= \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i} K^j(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) + \sum_{i:H_i \in \mathcal{S}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(\widetilde{U}_i, \widetilde{U}_i^{c^f}, \widehat{U}_i^{c^n}) = 42$$

Thus we get $V(\{H_1, H_3\}) = 42$.

For the grand coalition $\mathcal{H}$ the value of characteristic function is equal

$$V(\mathcal{H}) = \max_{U_i^c} \max_{U_i} \left(\sum_{i:H_i \in \mathcal{H}} \sum_{j \in \Gamma_i} K^j(U_i, U_i^c,) + \sum_{i:H_i \in \mathcal{H}} \sum_{j \in \Gamma_i^c} K_i^{c_j}(U_i, U_i^c) \right) = 83$$

In this example at this step we will use the solution with equal excess as an optimality principle.

$$\xi_{H_j} = V(H_i) + \frac{V(\mathcal{H}) - \sum\limits_{i \in \mathcal{H}} V(H_i)}{L}, \quad j = \overline{1, L}.$$

$$\xi_{H_1} = V(H_1) + \frac{V(\mathcal{H}) - (V(H_1) + V(H_2) + V(H_3))}{3} = 36.(3)$$

$$\xi_{H_2} = V(H_2) + \frac{V(\mathcal{H}) - (V(H_1) + V(H_2) + V(H_3))}{3} = 34.(3)$$

$$\xi_{H_3} = V(H_3) + \frac{V(\mathcal{H}) - (V(H_1) + V(H_2) + V(H_3))}{3} = 12.(3)$$

Second step. Now we need to solve three cooperative game as an optimality principle we will use proportional solution. For the game on hyperlink H_1 a characteristic function for players 1,2 and c_1

$$v^j(i) = \max_{u^i} \min_{U_j^c \bigcup U_j \backslash u^i} K^i(U_j, U_j^c).$$

$$v^j(c_i) = \max_{u^{c_i}} \min_{U_j^c \backslash u^{c_i} \bigcup U_j} K_j^{c_i}(U_j, U_j^c).$$

$$v^1(1) = 10, v^1(2) = 6, v^1(c_1) = 11$$

$$v^1(N_1) = \xi_{H_1} = 36.(3)$$

$$\mathcal{E}_1^1 = \frac{v^1(1)}{v^1(1) + v^1(2) + v^1(c_1)} v^1(N_1) = \frac{363.(3)}{27}$$

$$\mathcal{E}_2^1 = \frac{v^1(2)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{218}{27}$$

$$\mathcal{E}_{c_1}^1 = \frac{v^1(c)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{399.(6)}{27}$$

For the game on hyperlink H_2 a characteristic function for players 3, c_1 and c_2

$$v^2(3) = 15, v^2(c_1) = 8, v^2(c_2) = 11$$

$$v^2(N_2) = \xi_{H_2} = 34.(3)$$

$$\mathcal{E}_3^2 = \frac{v^1(1)}{v^1(1) + v^1(2) + v^1(c_1)} v^1(N_1) = \frac{515}{34}$$

$$\mathcal{E}_{c_1}^2 = \frac{v^1(2)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{274.(6)}{34}$$

$$\mathcal{E}_{c_2}^2 = \frac{v^1(c)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{377.(6)}{34}$$

For the game on hyperlink H_3 a characteristic function for players 4 and c_2

$$v^3(4) = 3, v^3(c_2) = 5$$

$$v^3(N_3) = \xi_{H_3} = 12.(3)$$

$$\mathcal{E}_4^3 = \frac{v^1(1)}{v^1(1) + v^1(2) + v^1(c_1)} v^1(N_1) = \frac{37}{8}$$

$$\mathcal{E}_{c_2}^3 = \frac{v^1(2)}{v^1(1) + v^1(2) + v^1(c)} v^1(N_1) = \frac{61.(6)}{8}$$

Third step. Now we need to sum the payoffs of players c_1 and c_2

$$\mathcal{E}_{c_1} = \sum_{j:H_j \in B_{c_1}} \mathcal{E}_{c_1}^j = \mathcal{E}_{c_1}^1 + \mathcal{E}_{c_1}^2 = \frac{399.(6)}{27} + \frac{274.(6)}{34}$$

$$\mathcal{E}_{c_2} = \sum_{j:H_j \in B_{c_2}} \mathcal{E}_{c_2}^j = \mathcal{E}_{c_2}^2 + \mathcal{E}_{c_2}^3 = \frac{377.(6)}{34} + \frac{61.(6)}{8}$$

So we get the imputation

$$\mathcal{E}_1 = \frac{363.(3)}{27}, \mathcal{E}_2 = \frac{218}{27}$$

$$\mathcal{E}_3 = \frac{515}{34}, \mathcal{E}_4 = \frac{37}{8}$$

$$\mathcal{E}_{c_1} = \frac{399.(6)}{27} + \frac{274.(6)}{34}, \mathcal{E}_{c_2} = \frac{377.(6)}{34} + \frac{61.(6)}{8}$$

Chapter 4. Software implementation

Finding the solution of the cooperative game is not easy and takes a lot of time. Because of that, I made a program to solve it. As an environment was chosen in Python 3. Python is a dynamically typed and high-level programming language. It is very useful for prototyping. Firstly it needed to create a class for hypergraphs.

```
class HyperGraph:
""""Undirected HyperGraph for game"""
def __init__(self, nodes: set, hyperlinks: dict):
    """
    Create new HyperGraph
    :param nodes: list of nodes
    :param hyperlinks: dict hyperlinks by name
    """
    self._nodes = nodes
    self._hyperlinks = hyperlinks
    self._incidence_graph = None
    self._check_correct_hyperlink()
    if self._incidence_graph is None:
        self._create_incidence_graph()
```

The initializer of this class takes as an input set of nodes and a dictionary of hyperlinks. Example of input:

```
graph = HyperGraph(
       {'v1', 'v2', 'v3', 'v4', 'vc'},
       {
           'h1': {'v1', 'v2', 'vc'},
           'h2': {'v3', 'v4', 'vc'},
       }
```

We already assume that hypergraph is should be acyclic and reduced. We automatically check these properties and is the hypergraph correctly defined.

```
def _check_correct_hyperlink(self):
    self._check_node_existence()
    self._check_cardinality()
    self._check_external_nodes()
    self._check_reduced()
    self._check_acycling()
```

For checking acyclicity it needs to create an incidence graph and check it for acyclicity. The incidence graph is a bipartite graph and to construct it and check we can use existing module NetworkX. Also, it will be useful to show this incidence graph by using module matplotlib.

```
    def _create_incidence_graph(self):
        g = nx.Graph()
        for player in self._nodes:
            g.add_node(player, bipartite=0)
        for name, hyperlink in self._hyperlinks.items():
            g.add_node(name, bipartite=1)
            for player in hyperlink:
                g.add_edge(name, player)
        self._incidence_graph = g
```

We suppose a special case of an original game where are all linked players have a bimatrix game between each other. Thus we defined a class of 2×2 bimatrix for each pair (i, j) in each hyperlink $H_j, i, j \in H_j$ with some methods.

```
class Bimatrix:
    """Bimaxtrix 2*2 for game"""
    def __init__(self, values: list):
        """
        Create new bimatrix
        :param values: [[(4, 8), (3, 6)], [(1, 3), (5, 6)]]
        """
        self._matrix = values
```

In docstrings shows the example of input. It is a list with two lists consisting payoffs for players (i, j).

Next step is define a class for calculating the values of characteristic function for any coalition (*_get_ ch_function_ hyperlinks*) of hyperlinks and imputation for them using the solution with equal excess (*_get_ imputation_ hyperlinks*) and the values of characteristic function for any players in each hyperlink (*_get_ ch_functions_ players*) and the total imputation for any players (*calculate_ imputations*) using proportional solution.

In the output, we want to get the values of the characteristic function for any coalition of hyperlinks, imputation for them, the values of the characteristic function for any player in each hyperlink, the total imputation for them and runtime.

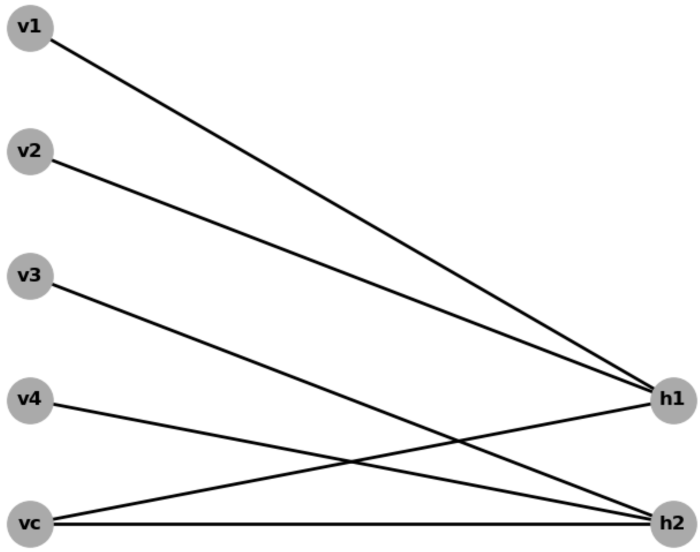
Now we will test this program with already calculated examples. The input for example in chapter 2

```
        graph = HyperGraph(
        {'v1', 'v2', 'v3', 'v4', 'vc'},
        {
            'h1': {'v1', 'v2', 'vc'},
            'h2': {'v3', 'v4', 'vc'},
        }
    )
    game = SimpleGame(
        graph,
        {
            ('v1', 'vc'): Bimatrix([[(4, 8), (3, 6)], [(1, 3), (5, 6)]]),
            ('v1', 'v2'): Bimatrix([[(6, 8), (6, 0)], [(4, 3), (0, 6)]]),
            ('v2', 'vc'): Bimatrix([[(3, 6), (5, 5)], [(0, 2), (4, 8)]]),

            ('v3', 'vc'): Bimatrix([[(8, 0), (6, 10)], [(3, 6), (9, 3)]]),
            ('v4', 'vc'): Bimatrix([[(5, 2), (8, 9)], [(7, 2), (6, 5)]]),
            ('v3', 'v4'): Bimatrix([[(0, 1), (10, 4)], [(7, 0), (3, 8)]]),
        }
    )
```

And the output is

```
v('h1',) = 33 (strategy: {'v1': 0, 'v2': 0, 'vc': 1})
v('h2',) = 31 (strategy: {'v3': 0, 'v4': 1, 'vc': 0})
v('h1', 'h2') = 74 (strategy: {'v1': 0, 'v2': 0, 'v3': 0, 'v4': 1, 'vc': 1})

ksi('h1') = 38.0
ksi('h2') = 36.0

v('v1' in h1) = 10
v('v2' in h1) = 6
v('vc' in h1) = 11
v('v3' in h2) = 15
v('v4' in h2) = 11
v('vc' in h2) = 8

eps('v1') = 14.074074074074074
eps('v2') = 8.444444444444445
eps('v3') = 15.882352941176471
eps('v4') = 11.647058823529411
```

```
eps('vc') = 23.952069716775597

Run time: 0.001001119613647461 seconds
```



The input for example in chapter 3

```
graph = HyperGraph(
    {'v1', 'v2', 'v3', 'v4', 'vc1', 'vc2'},
    {
        'h1': {'v1', 'v2', 'vc1'},
        'h2': {'v3', 'vc1', 'vc2'},
        'h3': {'v4', 'vc2'},
    }
)
game = SimpleGame(
    graph,
    {
        ('v1', 'vc1'): Bimatrix([[(4, 8), (3, 6)], [(1, 3), (5, 6)]]),
        ('v1', 'v2'): Bimatrix([[(6, 8), (6, 0)], [(4, 3), (0, 6)]]),
        ('v2', 'vc1'): Bimatrix([[(3, 6), (5, 5)], [(0, 2), (4, 8)]]),

        ('v3', 'vc1'): Bimatrix([[(8, 0), (6, 10)], [(3, 6), (9, 3)]]),
        ('v3', 'vc2'): Bimatrix([[(0, 1), (10, 4)], [(7, 0), (3, 8)]]),
        ('vc1', 'vc2'): Bimatrix([[(2, 5), (2, 7)], [(9, 8), (5, 6)]]),

        ('v4', 'vc2'): Bimatrix([[(1, 4), (2, 7)], [(4, 0), (3, 5)]]),
    }
)
```

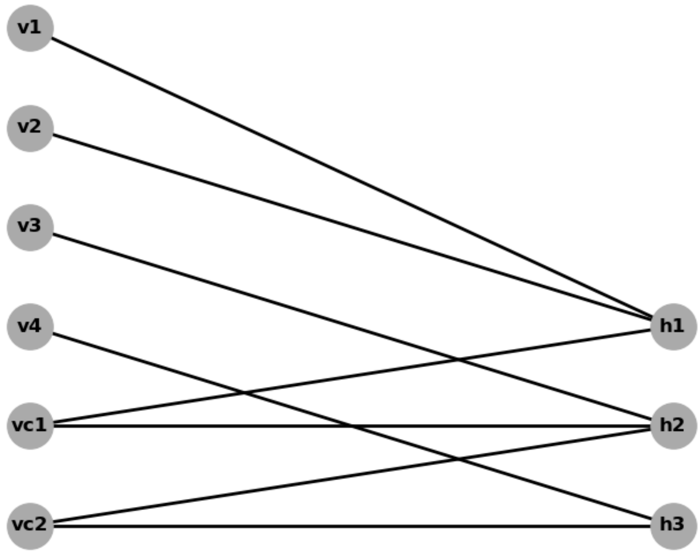
And the output is

```
v('h1',) = 33 (strategy: {'v1': 0, 'v2': 0, 'vc1': 1})
v('h2',) = 31 (strategy: {'v3': 0, 'vc1': 0, 'vc2': 1})
v('h3',) = 9 (strategy: {'v4': 0, 'vc2': 1})
v('h1', 'h2') = 74 (strategy: {'v1': 0, 'v2': 0, 'v3': 0, 'vc1': 1, 'vc2': 1})
v('h1', 'h3') = 42 (strategy: {'v1': 0, 'v2': 0, 'v4': 0, 'vc1': 1, 'vc2': 1})
v('h2', 'h3') = 40 (strategy: {'v3': 0, 'v4': 0, 'vc1': 0, 'vc2': 1})
v('h1', 'h2', 'h3') = 83 (strategy: {'v1': 0, 'v2': 0, 'v3': 0, 'v4': 0, 'vc1': 1, 'vc2': 1})

ksi('h1') = 36.333333333333336
ksi('h2') = 34.333333333333336
ksi('h3') = 12.333333333333334

v('v1' in h1) = 10
v('v2' in h1) = 6
v('vc1' in h1) = 11
v('v3' in h2) = 15
v('vc1' in h2) = 8
v('vc2' in h2) = 11
v('v4' in h3) = 3
v('vc2' in h3) = 5

eps('v1') = 13.456790123456791
eps('v2') = 8.074074074074074
eps('v3') = 15.147058823529411
eps('v4') = 4.625
eps('vc1') = 22.880900508351488
eps('vc2') = 18.81617647058824

Run time: 0.003995418548583984 seconds
```

As we see the output coincided with the obtained results. Now we will construct and solve by using the program two games with more complex communication structure.

Example 1. The first game we will construct based on theory from chapter two. Communication structure is given by a hypergraph with just 1 complex-player which included in all hyperlinks. The hypergraph is shown on fig. 9. Bimatrixes for all pairs of linked players were randomly generated and represent in output.

Output



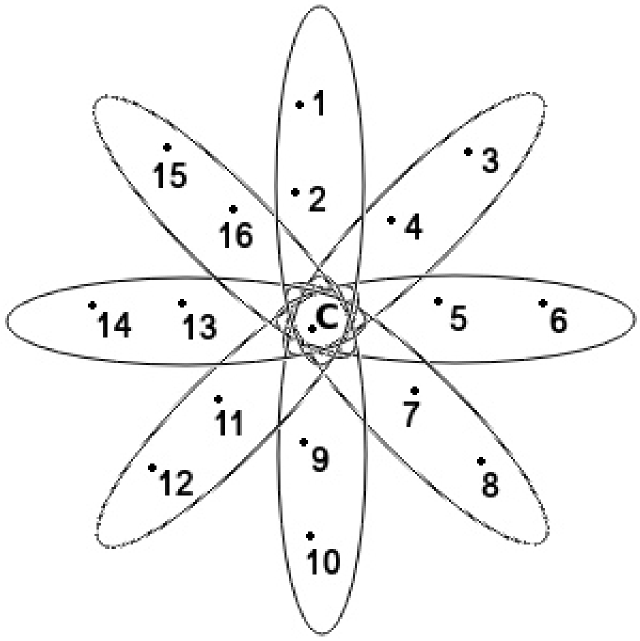


Figure 9

```
bimatrix[v01, v02] = [[(10, 0), (8, 7)], [(3, 7), (8, 5)]]
bimatrix[v01, vc] = [[(0, 0), (3, 8)], [(10, 2), (2, 8)]]
bimatrix[v02, vc] = [[(5, 9), (7, 9)], [(10, 8), (1, 7)]]
bimatrix[v03, v04] = [[(4, 1), (3, 10)], [(10, 4), (7, 8)]]
bimatrix[v03, vc] = [[(6, 10), (9, 3)], [(0, 5), (3, 4)]]
bimatrix[v04, vc] = [[(9, 3), (0, 9)], [(10, 6), (8, 3)]]
bimatrix[v05, v06] = [[(6, 3), (2, 3)], [(0, 6), (4, 3)]]
bimatrix[v05, vc] = [[(9, 7), (9, 2)], [(7, 10), (2, 3)]]
```

```
bimatrix[v06, vc] = [[(8, 4), (4, 6)], [(5, 0), (10, 8)]]
bimatrix[v07, v08] = [[(3, 1), (7, 5)], [(3, 0), (10, 9)]]
bimatrix[v07, vc] = [[(5, 2), (0, 10)], [(4, 0), (10, 10)]]
bimatrix[v08, vc] = [[(9, 1), (4, 3)], [(10, 4), (1, 0)]]
bimatrix[v09, v10] = [[(2, 0), (3, 6)], [(2, 8), (2, 7)]]
bimatrix[v09, vc] = [[(5, 0), (4, 6)], [(10, 10), (9, 3)]]
bimatrix[v10, vc] = [[(6, 9), (4, 0)], [(0, 10), (2, 4)]]
bimatrix[v11, v12] = [[(6, 9), (8, 0)], [(0, 4), (2, 1)]]
bimatrix[v11, vc] = [[(2, 6), (0, 4)], [(2, 0), (5, 0)]]
bimatrix[v12, vc] = [[(7, 8), (6, 3)], [(7, 1), (8, 10)]]
bimatrix[v13, v14] = [[(4, 7), (10, 9)], [(0, 4), (8, 3)]]
bimatrix[v13, vc] = [[(9, 5), (7, 6)], [(1, 2), (3, 4)]]
bimatrix[v14, vc] = [[(3, 4), (3, 5)], [(4, 8), (10, 6)]]
bimatrix[v15, v16] = [[(2, 4), (7, 8)], [(8, 1), (1, 7)]]
bimatrix[v15, vc] = [[(4, 3), (1, 1)], [(10, 4), (0, 9)]]
bimatrix[v16, vc] = [[(9, 0), (8, 9)], [(9, 3), (4, 10)]]

ksi('h1') = 43.0
ksi('h2') = 45.0
ksi('h3') = 37.0
ksi('h4') = 37.0
ksi('h5') = 45.0
ksi('h6') = 38.0
ksi('h7') = 45.0
ksi('h8') = 34.0

v('v01' in h1) = 10
v('v02' in h1) = 14
v('vc' in h1) = 16
v('v03' in h2) = 13
v('v04' in h2) = 16
v('vc' in h2) = 11
v('v05' in h3) = 11
v('v06' in h3) = 8
v('vc' in h3) = 13
v('v07' in h4) = 7
v('v08' in h4) = 9
v('vc' in h4) = 11
v('v09' in h5) = 11
```

```
v('v10' in h5) = 10
v('vc' in h5) = 15
v('v11' in h6) = 8
v('v12' in h6) = 11
v('vc' in h6) = 3
v('v13' in h7) = 11
v('v14' in h7) = 8
v('vc' in h7) = 9
v('v15' in h8) = 8
v('v16' in h8) = 15
v('vc' in h8) = 12

eps('v01') = 10.75
eps('v02') = 15.05
eps('v03') = 14.625
eps('v04') = 18.0
eps('v05') = 12.71875
eps('v06') = 9.25
eps('v07') = 9.592592592592593
eps('v08') = 12.333333333333334
eps('v09') = 13.75
eps('v10') = 12.5
eps('v11') = 13.818181818181818
eps('v12') = 19.0
eps('v13') = 17.678571428571427
eps('v14') = 12.857142857142858
eps('v15') = 7.771428571428571
eps('v16') = 14.571428571428571
eps('vc') = 109.73357082732083

Run time: 18.112125396728516 seconds
```

Values of characteristic function for all coalitions of hyperlinks added to the appendix.

Example 2. The second game we will construct based on theory from chapter three. The communication structure is given by the hypergraph which is shown on fig. 10. Bimatrixes for all pairs of linked players were randomly generated and represent in output.

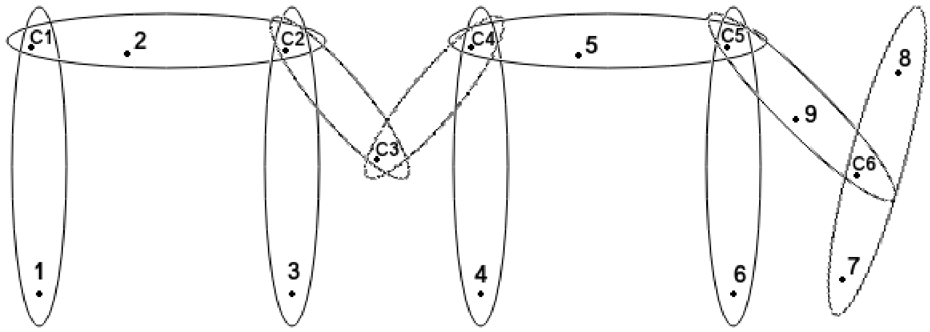


Figure 10

Output

```
bimatrix[v1, vc1] = [[(6, 2), (10, 7)], [(6, 10), (2, 9)]]
bimatrix[v2, vc1] = [[(10, 5), (6, 9)], [(4, 6), (1, 6)]]
bimatrix[v2, vc2] = [[(0, 4), (7, 7)], [(3, 3), (10, 5)]]
bimatrix[vc1, vc2] = [[(7, 3), (5, 2)], [(9, 4), (2, 10)]]
bimatrix[v3, vc2] = [[(10, 2), (5, 6)], [(0, 0), (6, 4)]]
bimatrix[vc2, vc3] = [[(8, 4), (5, 6)], [(2, 6), (4, 5)]]
bimatrix[vc3, vc4] = [[(7, 1), (3, 8)], [(3, 4), (2, 5)]]
bimatrix[v4, vc4] = [[(9, 3), (3, 2)], [(3, 7), (10, 9)]]
bimatrix[v5, vc4] = [[(6, 5), (8, 2)], [(5, 4), (0, 10)]]
bimatrix[v5, vc5] = [[(5, 2), (5, 1)], [(3, 2), (7, 0)]]
bimatrix[vc4, vc5] = [[(4, 4), (8, 1)], [(3, 6), (0, 9)]]
bimatrix[v6, vc5] = [[(2, 7), (8, 10)], [(8, 8), (4, 5)]]
bimatrix[v9, vc5] = [[(3, 5), (2, 8)], [(8, 6), (10, 9)]]
bimatrix[v9, vc6] = [[(8, 9), (1, 4)], [(7, 2), (9, 2)]]
bimatrix[vc5, vc6] = [[(7, 3), (1, 10)], [(3, 9), (6, 6)]]
bimatrix[v7, v8] = [[(3, 10), (10, 9)], [(1, 0), (10, 5)]]
bimatrix[v7, vc6] = [[(8, 7), (0, 4)], [(8, 7), (5, 7)]]
bimatrix[v8, vc6] = [[(1, 7), (0, 7)], [(8, 4), (2, 0)]]

ksi('h01') = 20.0
ksi('h02') = 35.0
ksi('h03') = 14.0
ksi('h04') = 11.0
ksi('h05') = 10.0
ksi('h06') = 22.0
ksi('h07') = 29.0
ksi('h08') = 21.0
```

```
ksi('h09') = 43.0
ksi('h10') = 32.0

v('v1' in h01) = 6
v('vc1' in h01) = 7
v('v2' in h02) = 9
v('vc1' in h02) = 11
v('vc2' in h02) = 8
v('v3' in h03) = 5
v('vc2' in h03) = 4
v('vc2' in h04) = 5
v('vc3' in h04) = 6
v('vc3' in h05) = 3
v('vc4' in h05) = 5
v('v4' in h06) = 9
v('vc4' in h06) = 3
v('v5' in h07) = 11
v('vc4' in h07) = 9
v('vc5' in h07) = 6
v('v6' in h08) = 8
v('vc5' in h08) = 8
v('v9' in h09) = 15
v('vc5' in h09) = 14
v('vc6' in h09) = 8
v('v7' in h10) = 8
v('v8' in h10) = 7
v('vc6' in h10) = 11

eps('v1') = 9.23076923076923
eps('v2') = 11.25
eps('v3') = 7.777777777777778
eps('v4') = 16.5
eps('v5') = 12.26923076923077
eps('v6') = 10.5
eps('v7') = 9.846153846153847
eps('v8') = 8.615384615384615
eps('v9') = 17.43243243243243
eps('vc1') = 24.51923076923077
eps('vc2') = 21.22222222222222
```

```
eps('vc3') = 9.75
eps('vc4') = 21.78846153846154
eps('vc5') = 33.46257796257797
eps('vc6') = 22.835758835758835

Run time: 26.648218154907227 seconds
```

Conclusion

As a result of the work carried out, a game-theoretic model of a cooperative game with a hypergraph as a communication structure. A new characteristic function for coalitions consisting of hyperlinks was introduced. A new allocation rule for this class of games was proposed.

A program algorithm in Python is created that provides a search for the values of the characteristic function for hyperlinks and their imputations using a solution with equal excess. Also, there are search and output values of the characteristic function for players in each hyperlink. The end result of the program is an imputation for all players of the original game. The program was successfully tested on new and already calculated examples.

In the future, it is planned to study other allocation rules for solving cooperative games with a communication structure given by a hypergraph.

References

[1] Myerson R. B. Graphs and cooperation in games // Math. Oper. Res. No 2. 1977. P. 225–229.

[2] Shapley LS A value for n-person games. Annals of Math. Studies 28, 1953. P. 307–317.

[3] Owen G Values of graph-restricted games. SIAM J. Alg. Disc. Meth. 7, 1986. P. 210-220.

[4] Meessen R. Communication games, Master's thesis. Department of Mathematics. University of Nijmegen, the Netherlands (in Dutch). 1988.

[5] Slikker M. A characterization of the position value // Int. J. Game Theory Vol. 33. 2005. P. 505–514.

[6] van den Nouwelandr A., Borm P, Tijs S Allocation rules for hypergraph communication situations // Int. J. Game Theory Vol. 20. 1992. P. 255–268.

[7] Shan E., G. Zhang, X. Shan The degree value for games with communication structure // Int. J. Game Theory Vol. 47. 2018. P. 857–871.

[8] Печерский С. Л., Яновская Е. Б. Кооперативные игры: решения и аксиомы. Европейский университет в Санкт-Петербурге, 2004. 459 с.

[9] Von Neumann J., Morgenstern O., Theory of Games and Economic Behavior. Princeton: Princeton University Press, 1994.

[10] Aumann R J, Myerson RB. Endogenous formation of links between players and of coalitions: an application of the Shapley value. In: The Shapley Value, Roth AE (ed.), Cambridge University Press, Cambridge, 1988. P. 175–191.

[11] Berge C. Hypergraphs. North-Holland, Mathematical library, pp. 1-3,1989. P. 155–162.

[12] Myerson R. B Conference structures and fair allocation rules. Intern. J. of Game Theory vol.9, 1980. P. 169–182.

[13] Casajus A. The position value is the Myerson value, in a sense. Int J Game Theory vol.36, 2007 P. 47-–55

Appendix

Source code Python 3.

```
import matplotlib.pyplot as plt
import networkx as nx
import warnings

from time import time
from itertools import combinations, product


class HyperGraph:
    """"Undirected HyperGraph for game"""
    def __init__(self, nodes: set, hyperlinks: dict):
        """
        Create new HyperGraph
        :param nodes: list of nodes
        :param hyperlinks: dict hyperlinks by name
        """
        self._nodes = nodes
        self._hyperlinks = hyperlinks
        self._incidence_graph = None
        self._check_correct_hyperlink()
        if self._incidence_graph is None:
            self._create_incidence_graph()

    def get_nodes(self):
        return self._nodes

    def get_hyperlink_names(self):
        return self._hyperlinks.keys()

    def get_hyperlink(self, name: str):
        return self._hyperlinks[name]

    def _create_incidence_graph(self):
        g = nx.Graph()
        for player in self._nodes:
            g.add_node(player, bipartite=0)
        for name, hyperlink in self._hyperlinks.items():
            g.add_node(name, bipartite=1)
            for player in hyperlink:
                g.add_edge(name, player)
        self._incidence_graph = g

    def show(self):
```

```
        warnings.filterwarnings('ignore')
        players, hyperlinks = nx.bipartite.sets(self._incidence_graph)
        pos = {}
        pos.update(
            (node, (1, index))
            for index, node in enumerate(sorted(players, reverse=True))
        )
        pos.update(
            (node, (2, index))
            for index, node in enumerate(sorted(hyperlinks, reverse=True))
        )
        nx.draw(
            self._incidence_graph, pos=pos,
            with_labels=True, font_weight='bold',
            node_color='#AAAAAA', node_size=800,
            width=2
        )
        plt.show()

    def _check_correct_hyperlink(self):
        self._check_node_existence()
        self._check_cardinality()
        self._check_external_nodes()
        self._check_reduced()
        self._check_acycling()

    def _check_node_existence(self):
        for hyperlink in self._hyperlinks.values():
            for node in hyperlink:
                if node not in self._nodes:
                    raise ValueError("{} doesn't exist".format(node))

    def _check_cardinality(self):
        for name, hyperlink in self._hyperlinks.items():
            if len(hyperlink) < 2:
                raise ValueError('{} cardinality less than 2'.format(name))

    def _check_external_nodes(self):
        all_nodes_in_hyperlinks = set()
        for hyperlink in self._hyperlinks.values():
            all_nodes_in_hyperlinks.update(hyperlink)
        external_nodes = self._nodes.difference(all_nodes_in_hyperlinks)
        if external_nodes:
            raise ValueError('{} is/are external'.format(external_nodes))

    def _check_reduced(self):
```

```
        for name1, hyperlink1 in self._hyperlinks.items():
            for name2, hyperlink2 in self._hyperlinks.items():
                if name1 != name2:
                    if hyperlink1.issubset(hyperlink2):
                        raise ValueError(
                            '{} include {}'.format(name2, name1)
                        )

    def _check_acycling(self):
        if self._incidence_graph is None:
            self._create_incidence_graph()
        cycles = nx.cycle_basis(self._incidence_graph)
        if cycles:
            raise ValueError(
                'incidence_graph has cycle: {}'.format(cycles[0])
            )


class Bimatrix:
    """Bimaxtrix 2*2 for game"""
    def __init__(self, values: list):
        """
        Create new bimatrix
        :param values: [[(4, 8), (3, 6)], [(1, 3), (5, 6)]]
        """
        self._matrix = values

    def __getitem__(self, item):
        """Get element of matrix by [(i, j)] instead of [i][j]"""
        return self._matrix[item[0]][item[1]]

    def transpose(self):
        """Transpose matrix and swap payoffs"""
        matrix_T = [
            [(j, i) for i, j in row]
            for row in self._matrix
        ]
        matrix_T[1][0], matrix_T[0][1] = matrix_T[0][1], matrix_T[1][0]
        return Bimatrix(matrix_T)

    def maxmin_minmax(self):
        """Get the payoff using max min"""
        rows = [
            [i[0] for i in row]
            for row in self._matrix
        ]
```

```
        row = 0 if min(rows[0]) >= min(rows[1]) else 1
        cols = [
            [col[i] for col in rows]
            for i in range(len(rows[0]))
        ]
        col = 0 if max(cols[0]) <= max(cols[1]) else 1
        return rows[row][col]


class SimpleGame:
    def __init__(self, graph: HyperGraph, bimatrix: dict):
        """
        Create new SimpleGame
        :param graph: Undirected HyperGraph
        :param bimatrix: Bimatrix 2*2
        """
        self._graph = graph
        self._bimatrix = bimatrix
        self._check_correct_bimatrix()

    def _check_correct_bimatrix(self):
        for name in self._graph.get_hyperlink_names():
            hyperlink = sorted(self._graph.get_hyperlink(name))
            for node1, node2 in combinations(hyperlink, 2):
                if (node1, node2) not in self._bimatrix:
                    raise ValueError(
                        "Bimatrix for ({}, {}) doesn’t exist".format(
                            node1, node2
                        )
                    )

    def get_all_coalitions(self):
        all_s = []
        hyperlink_names = self._graph.get_hyperlink_names()
        for i in range(1, len(hyperlink_names) + 1):
            all_s.extend(combinations(hyperlink_names, i))
        return all_s

    def get_complement(self, coalition: tuple):
        return set(self._graph.get_hyperlink_names()).difference(
            set(coalition)
        )

    def _get_max_function_complement(self):
        result = {}
        for s in self.get_all_coalitions():
```

```
            n_s = self.get_complement(s)
            all_pairs = set()
            all_players = set()

            for some_n_s in n_s:
                for players in combinations(
                        sorted(self._graph.get_hyperlink(some_n_s)), 2
                ):
                    all_pairs.add(players)
                    all_players.update(players)
            all_players = {
                i: j
                for i, j in zip(sorted(all_players), range(len(all_players)))
            }

            max_value = -float('inf')
            max_strategy = None
            for strategy in product(range(2), repeat=len(all_players)):
                current_value = 0
                for player1, player2 in all_pairs:
                    index1 = strategy[all_players[player1]]
                    index2 = strategy[all_players[player2]]
                    current_value += sum(
                        self._bimatrix[(player1, player2)][index1, index2]
                    )
                if current_value > max_value:
                    max_value = current_value
                    max_strategy = strategy

            result[s] = {
                i: j
                for i, j in zip(sorted(all_players), max_strategy)
            }

        return result

    def _get_ch_function_hyperlinks(self, strategies_by_coalition):
        result = {}
        for s in self.get_all_coalitions():
            fixed_strategy = strategies_by_coalition[s]
            all_pairs = set()
            all_players = set()

            for some_s in s:
                for players in combinations(
                        sorted(self._graph.get_hyperlink(some_s)), 2
```

```
                ):
                    all_pairs.add(players)
                    all_players.update(players)
            all_players = {
                i: j
                for i, j in zip(sorted(all_players), range(len(all_players)))
            }

            max_value = -float('inf')
            max_strategy = None
            for strategy in product(range(2), repeat=len(all_players)):

                check_good_strategy = True
                for player, player_position in all_players.items():
                    if player in fixed_strategy \
                       and fixed_strategy[player] != strategy[player_position]:
                        check_good_strategy = False
                        break
                if not check_good_strategy:
                    continue

                current_value = 0
                for player1, player2 in all_pairs:
                    index1 = strategy[all_players[player1]]
                    index2 = strategy[all_players[player2]]
                    current_value += sum(
                        self._bimatrix[(player1, player2)][index1, index2]
                    )
                if current_value > max_value:
                    max_value = current_value
                    max_strategy = strategy

            result[s] = (
                max_value,
                {
                    i: j
                    for i, j in zip(sorted(all_players), max_strategy)
                }
            )
        return result

    def _get_imputation_hyperlinks(self, ch_functions):
        fraction = ch_functions[
            tuple(sorted(self._graph.get_hyperlink_names()))
        ][0]
        v_s = {}
```

```
        for s, (value, _) in ch_functions.items():
            if len(s) == 1:
                v_s[s[0]] = value
        fraction -= sum(v_s.values())
        fraction /= len(v_s)
        return {
            s: ksi + fraction
            for s, ksi in v_s.items()
        }

    def _get_ch_functions_players(self):
        result = {}
        for hyperlink in self._graph.get_hyperlink_names():
            result[hyperlink] = {}
            pairs_by_player = {}

            for player1, player2 in combinations(
                    sorted(self._graph.get_hyperlink(hyperlink)), 2
            ):
                if player1 not in pairs_by_player:
                    pairs_by_player[player1] = set()
                if player2 not in pairs_by_player:
                    pairs_by_player[player2] = set()
                pairs_by_player[player1].add((player1, player2))
                pairs_by_player[player2].add((player1, player2))

            for player, pairs in pairs_by_player.items():
                result[hyperlink][player] = 0
                for player1, player2 in pairs:
                    bimatrix = self._bimatrix[(player1, player2)]
                    if player == player2:
                        bimatrix = bimatrix.transpose()
                    result[hyperlink][player] += bimatrix.maxmin_minmax()

        return result

    def _get_imputation_players(self, ch_functions, ksi):
        result = {}
        for hyperlink_name, chis in ch_functions.items():
            sum_chi = sum(
                chis.values()
            )
            for player, chi in chis.items():
                if player not in result:
                    result[player] = 0
                result[player] += chi * ksi[hyperlink_name] / sum_chi
```

```
        return result

    def calculate_imputations(self):
        fixed_strategies = self._get_max_function_complement()
        ch_functions_hyperlinks = self._get_ch_function_hyperlinks(
            fixed_strategies
        )
        ksi = self._get_imputation_hyperlinks(ch_functions_hyperlinks)
        ch_functions_players = self._get_ch_functions_players()
        eps = self._get_imputation_players(ch_functions_players, ksi)
        return ch_functions_hyperlinks, ksi, ch_functions_players, eps

    def calculate_and_print_report(self):
        start_time = time()
        (
            ch_functions_hyperlinks, imputations_hyperlinks,
            ch_functions_players, imputations_players
        ) = self.calculate_imputations()

        for coalition, (value, strategies) in ch_functions_hyperlinks.items():
            print(
                'v{} = {} (strategy: {})'.format(coalition, value, strategies))
        print()
        for s, ksi in imputations_hyperlinks.items():
            print("ksi('{}') = {}".format(s, ksi))
        print()
        for coalition, info in ch_functions_players.items():
            for player, value in info.items():
                print("v('{}' in {}) = {}".format(player, coalition, value))
        print()
        for player, eps in sorted(imputations_players.items()):
            print("eps('{}') = {}".format(player, eps))
        print('\nRun time: {} seconds'.format(time() - start_time))


if __name__ == '__main__':
    graph = HyperGraph(
        {'v1', 'v2', 'v3', 'v4', 'vc'},
        {
            'h1': {'v1', 'v2', 'vc'},
            'h2': {'v3', 'v4', 'vc'},
        }
    )
    game = SimpleGame(
        graph,
        {
```

```
            ('v1', 'vc'): Bimatrix([[(4, 8), (3, 6)], [(1, 3), (5, 6)]]),
            ('v1', 'v2'): Bimatrix([[(6, 8), (6, 0)], [(4, 3), (0, 6)]]),
            ('v2', 'vc'): Bimatrix([[(3, 6), (5, 5)], [(0, 2), (4, 8)]]),

            ('v3', 'vc'): Bimatrix([[(8, 0), (6, 10)], [(3, 6), (9, 3)]]),
            ('v4', 'vc'): Bimatrix([[(5, 2), (8, 9)], [(7, 2), (6, 5)]]),
            ('v3', 'v4'): Bimatrix([[(0, 1), (10, 4)], [(7, 0), (3, 8)]]),
        }
    )

    # graph = HyperGraph(
    #     {'v1', 'v2', 'v3', 'v4', 'vc1', 'vc2'},
    #     {
    #         'h1': {'v1', 'v2', 'vc1'},
    #         'h2': {'v3', 'vc1', 'vc2'},
    #         'h3': {'v4', 'vc2'},
    #     }
    # )
    # game = SimpleGame(
    #     graph,
    #     {
    #         ('v1', 'vc1'): Bimatrix([[(4, 8), (3, 6)], [(1, 3), (5, 6)]]),
    #         ('v1', 'v2'): Bimatrix([[(6, 8), (6, 0)], [(4, 3), (0, 6)]]),
    #         ('v2', 'vc1'): Bimatrix([[(3, 6), (5, 5)], [(0, 2), (4, 8)]]),
    #
    #         ('v3', 'vc1'): Bimatrix([[(8, 0), (6, 10)], [(3, 6), (9, 3)]]),
    #         ('v3', 'vc2'): Bimatrix([[(0, 1), (10, 4)], [(7, 0), (3, 8)]]),
    #         ('vc1', 'vc2'): Bimatrix([[(2, 5), (2, 7)], [(9, 8), (5, 6)]]),
    #
    #         ('v4', 'vc2'): Bimatrix([[(1, 4), (2, 7)], [(4, 0), (3, 5)]]),
    #     }
    # )

    graph.show()
    game.calculate_and_print_report()
```

Values of characteristic function for all coalitions of hyperlinks for example 1.

```
v('h1',) = 43
v('h2',) = 45
v('h3',) = 37
v('h4',) = 37
v('h5',) = 45
v('h6',) = 38
v('h7',) = 45
v('h8',) = 34
```

```
v('h1', 'h2') = 88
v('h1', 'h3') = 80
v('h1', 'h4') = 80
v('h1', 'h5') = 88
v('h1', 'h6') = 81
v('h1', 'h7') = 88
v('h1', 'h8') = 77
v('h2', 'h3') = 82
v('h2', 'h4') = 82
v('h2', 'h5') = 90
v('h2', 'h6') = 83
v('h2', 'h7') = 90
v('h2', 'h8') = 79
v('h3', 'h4') = 74
v('h3', 'h5') = 82
v('h3', 'h6') = 75
v('h3', 'h7') = 82
v('h3', 'h8') = 71
v('h4', 'h5') = 82
v('h4', 'h6') = 75
v('h4', 'h7') = 82
v('h4', 'h8') = 71
v('h5', 'h6') = 83
v('h5', 'h7') = 90
v('h5', 'h8') = 79
v('h6', 'h7') = 83
v('h6', 'h8') = 72
v('h7', 'h8') = 79
v('h1', 'h2', 'h3') = 125
v('h1', 'h2', 'h4') = 125
v('h1', 'h2', 'h5') = 133
v('h1', 'h2', 'h6') = 126
v('h1', 'h2', 'h7') = 133
v('h1', 'h2', 'h8') = 122
v('h1', 'h3', 'h4') = 117
v('h1', 'h3', 'h5') = 125
v('h1', 'h3', 'h6') = 118
v('h1', 'h3', 'h7') = 125
v('h1', 'h3', 'h8') = 114
v('h1', 'h4', 'h5') = 125
v('h1', 'h4', 'h6') = 118
v('h1', 'h4', 'h7') = 125
v('h1', 'h4', 'h8') = 114
v('h1', 'h5', 'h6') = 126
v('h1', 'h5', 'h7') = 133
v('h1', 'h5', 'h8') = 122
```

```
v('h1', 'h6', 'h7') = 126
v('h1', 'h6', 'h8') = 115
v('h1', 'h7', 'h8') = 122
v('h2', 'h3', 'h4') = 119
v('h2', 'h3', 'h5') = 127
v('h2', 'h3', 'h6') = 120
v('h2', 'h3', 'h7') = 127
v('h2', 'h3', 'h8') = 116
v('h2', 'h4', 'h5') = 127
v('h2', 'h4', 'h6') = 120
v('h2', 'h4', 'h7') = 127
v('h2', 'h4', 'h8') = 116
v('h2', 'h5', 'h6') = 128
v('h2', 'h5', 'h7') = 135
v('h2', 'h5', 'h8') = 124
v('h2', 'h6', 'h7') = 128
v('h2', 'h6', 'h8') = 117
v('h2', 'h7', 'h8') = 124
v('h3', 'h4', 'h5') = 119
v('h3', 'h4', 'h6') = 112
v('h3', 'h4', 'h7') = 119
v('h3', 'h4', 'h8') = 108
v('h3', 'h5', 'h6') = 120
v('h3', 'h5', 'h7') = 127
v('h3', 'h5', 'h8') = 116
v('h3', 'h6', 'h7') = 120
v('h3', 'h6', 'h8') = 109
v('h3', 'h7', 'h8') = 116
v('h4', 'h5', 'h6') = 120
v('h4', 'h5', 'h7') = 127
v('h4', 'h5', 'h8') = 116
v('h4', 'h6', 'h7') = 120
v('h4', 'h6', 'h8') = 109
v('h4', 'h7', 'h8') = 116
v('h5', 'h6', 'h7') = 128
v('h5', 'h6', 'h8') = 117
v('h5', 'h7', 'h8') = 124
v('h6', 'h7', 'h8') = 117
v('h1', 'h2', 'h3', 'h4') = 162
v('h1', 'h2', 'h3', 'h5') = 170
v('h1', 'h2', 'h3', 'h6') = 163
v('h1', 'h2', 'h3', 'h7') = 170
v('h1', 'h2', 'h3', 'h8') = 159
v('h1', 'h2', 'h4', 'h5') = 170
v('h1', 'h2', 'h4', 'h6') = 163
v('h1', 'h2', 'h4', 'h7') = 170
```

```
v('h1', 'h2', 'h4', 'h8') = 159
v('h1', 'h2', 'h5', 'h6') = 130
v('h1', 'h2', 'h5', 'h7') = 178
v('h1', 'h2', 'h5', 'h8') = 167
v('h1', 'h2', 'h6', 'h7') = 171
v('h1', 'h2', 'h6', 'h8') = 160
v('h1', 'h2', 'h7', 'h8') = 167
v('h1', 'h3', 'h4', 'h5') = 162
v('h1', 'h3', 'h4', 'h6') = 155
v('h1', 'h3', 'h4', 'h7') = 162
v('h1', 'h3', 'h4', 'h8') = 151
v('h1', 'h3', 'h5', 'h6') = 163
v('h1', 'h3', 'h5', 'h7') = 170
v('h1', 'h3', 'h5', 'h8') = 159
v('h1', 'h3', 'h6', 'h7') = 163
v('h1', 'h3', 'h6', 'h8') = 152
v('h1', 'h3', 'h7', 'h8') = 159
v('h1', 'h4', 'h5', 'h6') = 163
v('h1', 'h4', 'h5', 'h7') = 170
v('h1', 'h4', 'h5', 'h8') = 159
v('h1', 'h4', 'h6', 'h7') = 163
v('h1', 'h4', 'h6', 'h8') = 152
v('h1', 'h4', 'h7', 'h8') = 159
v('h1', 'h5', 'h6', 'h7') = 171
v('h1', 'h5', 'h6', 'h8') = 160
v('h1', 'h5', 'h7', 'h8') = 167
v('h1', 'h6', 'h7', 'h8') = 160
v('h2', 'h3', 'h4', 'h5') = 164
v('h2', 'h3', 'h4', 'h6') = 157
v('h2', 'h3', 'h4', 'h7') = 164
v('h2', 'h3', 'h4', 'h8') = 153
v('h2', 'h3', 'h5', 'h6') = 127
v('h2', 'h3', 'h5', 'h7') = 172
v('h2', 'h3', 'h5', 'h8') = 161
v('h2', 'h3', 'h6', 'h7') = 165
v('h2', 'h3', 'h6', 'h8') = 154
v('h2', 'h3', 'h7', 'h8') = 161
v('h2', 'h4', 'h5', 'h6') = 165
v('h2', 'h4', 'h5', 'h7') = 172
v('h2', 'h4', 'h5', 'h8') = 161
v('h2', 'h4', 'h6', 'h7') = 165
v('h2', 'h4', 'h6', 'h8') = 154
v('h2', 'h4', 'h7', 'h8') = 161
v('h2', 'h5', 'h6', 'h7') = 173
v('h2', 'h5', 'h6', 'h8') = 162
v('h2', 'h5', 'h7', 'h8') = 169
```

```
v('h2', 'h6', 'h7', 'h8') = 162
v('h3', 'h4', 'h5', 'h6') = 157
v('h3', 'h4', 'h5', 'h7') = 164
v('h3', 'h4', 'h5', 'h8') = 153
v('h3', 'h4', 'h6', 'h7') = 157
v('h3', 'h4', 'h6', 'h8') = 146
v('h3', 'h4', 'h7', 'h8') = 153
v('h3', 'h5', 'h6', 'h7') = 165
v('h3', 'h5', 'h6', 'h8') = 154
v('h3', 'h5', 'h7', 'h8') = 161
v('h3', 'h6', 'h7', 'h8') = 154
v('h4', 'h5', 'h6', 'h7') = 165
v('h4', 'h5', 'h6', 'h8') = 154
v('h4', 'h5', 'h7', 'h8') = 161
v('h4', 'h6', 'h7', 'h8') = 154
v('h5', 'h6', 'h7', 'h8') = 162
v('h1', 'h2', 'h3', 'h4', 'h5') = 207
v('h1', 'h2', 'h3', 'h4', 'h6') = 200
v('h1', 'h2', 'h3', 'h4', 'h7') = 207
v('h1', 'h2', 'h3', 'h4', 'h8') = 196
v('h1', 'h2', 'h3', 'h5', 'h6') = 164
v('h1', 'h2', 'h3', 'h5', 'h7') = 215
v('h1', 'h2', 'h3', 'h5', 'h8') = 204
v('h1', 'h2', 'h3', 'h6', 'h7') = 208
v('h1', 'h2', 'h3', 'h6', 'h8') = 197
v('h1', 'h2', 'h3', 'h7', 'h8') = 204
v('h1', 'h2', 'h4', 'h5', 'h6') = 170
v('h1', 'h2', 'h4', 'h5', 'h7') = 215
v('h1', 'h2', 'h4', 'h5', 'h8') = 204
v('h1', 'h2', 'h4', 'h6', 'h7') = 208
v('h1', 'h2', 'h4', 'h6', 'h8') = 197
v('h1', 'h2', 'h4', 'h7', 'h8') = 204
v('h1', 'h2', 'h5', 'h6', 'h7') = 178
v('h1', 'h2', 'h5', 'h6', 'h8') = 165
v('h1', 'h2', 'h5', 'h7', 'h8') = 212
v('h1', 'h2', 'h6', 'h7', 'h8') = 205
v('h1', 'h3', 'h4', 'h5', 'h6') = 200
v('h1', 'h3', 'h4', 'h5', 'h7') = 207
v('h1', 'h3', 'h4', 'h5', 'h8') = 196
v('h1', 'h3', 'h4', 'h6', 'h7') = 200
v('h1', 'h3', 'h4', 'h6', 'h8') = 189
v('h1', 'h3', 'h4', 'h7', 'h8') = 196
v('h1', 'h3', 'h5', 'h6', 'h7') = 208
v('h1', 'h3', 'h5', 'h6', 'h8') = 197
v('h1', 'h3', 'h5', 'h7', 'h8') = 204
v('h1', 'h3', 'h6', 'h7', 'h8') = 197
```

```
v('h1', 'h4', 'h5', 'h6', 'h7') = 208
v('h1', 'h4', 'h5', 'h6', 'h8') = 197
v('h1', 'h4', 'h5', 'h7', 'h8') = 204
v('h1', 'h4', 'h6', 'h7', 'h8') = 197
v('h1', 'h5', 'h6', 'h7', 'h8') = 205
v('h2', 'h3', 'h4', 'h5', 'h6') = 202
v('h2', 'h3', 'h4', 'h5', 'h7') = 209
v('h2', 'h3', 'h4', 'h5', 'h8') = 198
v('h2', 'h3', 'h4', 'h6', 'h7') = 202
v('h2', 'h3', 'h4', 'h6', 'h8') = 191
v('h2', 'h3', 'h4', 'h7', 'h8') = 198
v('h2', 'h3', 'h5', 'h6', 'h7') = 210
v('h2', 'h3', 'h5', 'h6', 'h8') = 162
v('h2', 'h3', 'h5', 'h7', 'h8') = 206
v('h2', 'h3', 'h6', 'h7', 'h8') = 199
v('h2', 'h4', 'h5', 'h6', 'h7') = 210
v('h2', 'h4', 'h5', 'h6', 'h8') = 199
v('h2', 'h4', 'h5', 'h7', 'h8') = 206
v('h2', 'h4', 'h6', 'h7', 'h8') = 199
v('h2', 'h5', 'h6', 'h7', 'h8') = 207
v('h3', 'h4', 'h5', 'h6', 'h7') = 202
v('h3', 'h4', 'h5', 'h6', 'h8') = 191
v('h3', 'h4', 'h5', 'h7', 'h8') = 198
v('h3', 'h4', 'h6', 'h7', 'h8') = 191
v('h3', 'h5', 'h6', 'h7', 'h8') = 199
v('h4', 'h5', 'h6', 'h7', 'h8') = 199
v('h1', 'h2', 'h3', 'h4', 'h5', 'h6') = 204
v('h1', 'h2', 'h3', 'h4', 'h5', 'h7') = 252
v('h1', 'h2', 'h3', 'h4', 'h5', 'h8') = 241
v('h1', 'h2', 'h3', 'h4', 'h6', 'h7') = 245
v('h1', 'h2', 'h3', 'h4', 'h6', 'h8') = 234
v('h1', 'h2', 'h3', 'h4', 'h7', 'h8') = 241
v('h1', 'h2', 'h3', 'h5', 'h6', 'h7') = 212
v('h1', 'h2', 'h3', 'h5', 'h6', 'h8') = 199
v('h1', 'h2', 'h3', 'h5', 'h7', 'h8') = 249
v('h1', 'h2', 'h3', 'h6', 'h7', 'h8') = 242
v('h1', 'h2', 'h4', 'h5', 'h6', 'h7') = 253
v('h1', 'h2', 'h4', 'h5', 'h6', 'h8') = 242
v('h1', 'h2', 'h4', 'h5', 'h7', 'h8') = 249
v('h1', 'h2', 'h4', 'h6', 'h7', 'h8') = 242
v('h1', 'h2', 'h5', 'h6', 'h7', 'h8') = 250
v('h1', 'h3', 'h4', 'h5', 'h6', 'h7') = 245
v('h1', 'h3', 'h4', 'h5', 'h6', 'h8') = 234
v('h1', 'h3', 'h4', 'h5', 'h7', 'h8') = 241
v('h1', 'h3', 'h4', 'h6', 'h7', 'h8') = 234
v('h1', 'h3', 'h5', 'h6', 'h7', 'h8') = 242
```

```
v('h1', 'h4', 'h5', 'h6', 'h7', 'h8') = 242
v('h2', 'h3', 'h4', 'h5', 'h6', 'h7') = 247
v('h2', 'h3', 'h4', 'h5', 'h6', 'h8') = 236
v('h2', 'h3', 'h4', 'h5', 'h7', 'h8') = 243
v('h2', 'h3', 'h4', 'h6', 'h7', 'h8') = 236
v('h2', 'h3', 'h5', 'h6', 'h7', 'h8') = 244
v('h2', 'h4', 'h5', 'h6', 'h7', 'h8') = 244
v('h3', 'h4', 'h5', 'h6', 'h7', 'h8') = 236
v('h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'h7') = 252
v('h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'h8') = 239
v('h1', 'h2', 'h3', 'h4', 'h5', 'h7', 'h8') = 286
v('h1', 'h2', 'h3', 'h4', 'h6', 'h7', 'h8') = 279
v('h1', 'h2', 'h3', 'h5', 'h6', 'h7', 'h8') = 247
v('h1', 'h2', 'h4', 'h5', 'h6', 'h7', 'h8') = 287
v('h1', 'h3', 'h4', 'h5', 'h6', 'h7', 'h8') = 279
v('h2', 'h3', 'h4', 'h5', 'h6', 'h7', 'h8') = 281
v('h1', 'h2', 'h3', 'h4', 'h5', 'h6', 'h7', 'h8') = 324
```