University of Bergamo

Department of Mathematics, Statistics,
Computer Science and Applications

Computational methods for financial and economic forecasting and decisions (XXIII° Cycle)

Voting Cohesions and Collusions via Cooperative Games

PhD student:
Angelo Uristani

Supervisor:
Prof. Gianfranco Gambarelli

S.S.D.: SECS-S/06 METODI MATEMATICI DELL'ECONOMIA E DELLE SCIENZE ATTUARIALI E FINANZIARIE

Contents

Introduction

In this thesis we present some original contributions to the Cooperative Game Theory with applications in the economic and financial fields. Such contributions represent a continuation of my studies undertaken both in the second level degree and during subsequent activities that led me to produce publications in collaboration with other authors.

The first contribution (presented in Part 1) concerns the development of a model for the analysis of corporate control. The model allows to measure the influence of each investor in every company taking into account the different types of companies and the relationships among the investors.

This work is based on an extension of the multicameral cohesion voting games developed in (Gambarelli and Uristani, 2009) and it expands the related economic applications, some of which have already been presented in (Uristani, 2007).

After defining the model we present three algorithms for the computations in order to guarantee a trade-off between the set of the cases that can be analyzed and the computational time. It is important to reduce the computational time since in such way it is possible to obtain the results in a reasonable time even in cases in which there are a number of companies and investors.

Apart from the analysis of corporate control, the model can be applied for other purposes; in particular, we consider three applications.

The first one concerns the definition of a strategy for an investor who is interested to obtain a certain level of influence in a target company. The strategy consists in the purchase of a certain amount of shares of different companies in order to obtain the desired level of influence while minimizing the costs. We provide an algorithm for the computation of such strategy.

The second one concerns the identification of the investors who have a dominant influence in a company; this is important for the authorities for the control of the regulated markets because special rules apply to these investors. We provide an algorithm also for this application.

Finally we propose an application of the model to Corporate Finance.

The first part of the thesis concludes with the presentation of some possible developments of the model.

The second contribution (presented in Part 2) is related to the analysis of scoring methods that are robust to collusion. In Economics these methods are useful when the evaluation of a good requires that some experts provide an estimate that may be, at least in part, subjective. In such cases there may be some collusion among the experts detectable by these methods.

A remarkable application of Collusion-Robust methods is the determination of the Euribor rate since it is computed on the basis of the evaluations provided by a panel of banks.

We also present two Collusion-Robust methods: the Coherent Majority Average introduced in (Gambarelli, 2008) and the Anti-Collusion Average in (Bertini, Gambarelli and Uristani, 2010).

Since there is more than one method, it is necessary to decide which one to apply. The original contribution concerns the development of an analysis method, based on Cooperative Game Theory, that allows to verify which Collusion-Robust scoring method is the most suitable for a given situation.

PART 1
VOTING GAMES

The contribution of this part of the thesis concerns the development of a model for the analysis of corporate control. There are several contributions in literature on this subject; see, for instance, (Gambarelli and Owen, 1994) for a model based on Game Theory and (Crama and Leruth, 2007) and (Levy, 2007) for a survey of other models that have been developed.

As far as we know there is not a model that allows considering jointly:

- different types of companies (f. i., the Italian "società cooperative");
- different relationships among the investors (f. i., voting agreements);
- different types of shares (f. i., golden shares).

The model that we present here allows considering such cases.

In the first chapter we provide some definitions and some models of Game Theory that are necessary for the comprehension of the original results.

In the second chapter we present a model introduced in (Gambarelli and Uristani, 2009) that is necessary as a base to develop the general model.

In the third chapter the new model is presented along with some remarks about the assumptions that have been made.

In the fourth chapter we develop three algorithms that can be used to apply the model to specific cases.

In the fifth chapter we show some applications including the identification, for each investor, of the amount of shares to buy in every company in order to reach a certain degree of control in a target company.

Finally, in the sixth chapter the first part of the thesis is concluded with some considerations on possible further developments.

CHAPTER 1
Pre-existing situation

In this chapter some definitions necessary to the comprehension of the thesis are provided; for those who are interested to take a deeper look into Game Theory, further information can be found in some textbooks on the subject. We suggest (Owen, 1995), (Gambarelli, 2003) and (Slikker and Van Den Nouweland, 2000) in addition to the classic (von Neumann and Morgenstern, 1944).

Let $N = \{1, \ldots, n\}$ be the set of such agents that, for the sake of convention, are called players. There exist different representations of a game: strategic, extensive and in characteristic function form. In this thesis we adopt only the representation in characteristic function form.

A **game in characteristic function form** (N, v) is a game described by a characteristic function v that assigns a value to each coalition $S \subseteq N$. In this thesis we consider only TU-Games.

A **simple game** is a game in characteristic function form in which the function $v(S)$ can take only the values 0 or 1. The coalition S is winning if $v(S) = 1$, otherwise it is losing.

A player is called **crucial** for a coalition if such coalition is winning with him and is losing without him.

Simple games are suited to represent voting situations in which it is important to know if a player belongs to winning coalitions or to losing coalitions; in such a way they can be used to represent the decision process of an assembly. Assume for example that the weight of the i-th player in such assembly is $w(i)$ and fix a majority quota $q \in R$ (where $\frac{1}{2}\sum_{i \in N} w(i) < q \leq \sum_{i \in N} w(i)$) that is necessary to approve a resolution.

A **weighted majority game** is a simple game in which for each coalition S:

- $v(S) = 1$ if $\sum_{i \in S} w(i) \geq q$;
- $v(S) = 0$ in the other cases.

For all games in characteristic function form, an **imputation** is an allocation, represented by a vector $(x_1, \ldots, x_n)$, that respects the conditions of individual rationality and efficiency. An allocation satisfies the condition of individual rationality if it doesn't assign to any player a payoff lower than the one he would obtain alone and it satisfies the condition of efficiency if it gives to the players the whole value of the global coalition.

1.1 Legal references

In order to create a model as close as possible to reality, we take the cue from the Italian legislation; see for reference (Gambino et al., 2006) and (Annunziata, 2004). However it is necessary to clarify that the model can also be used for other countries; in particular, a similar legislation exists in other countries that belong to the European Union and in many developed countries outside the EU. Finally, in this thesis we will also refer to other laws applied outside the EU.

First, it is important to note that the members of a company may decide to coordinate their voting behavior through some agreements. In particular, in this thesis we refer to the voting agreements through which a group of members can exercise the voting rights in a coordinated way.

To apply the model it is important to note that there exist cases in which someone who hold some shares with voting rights cannot exercise them due to some regulations. These limits on the exercise of voting rights vary depending on the considered legislation. In this thesis only the shares or quotas of ownership for which it is possible to exercise the voting rights are considered, unless otherwise specified.

Finally it is necessary to mention the concept of **tender offer**. In Italy such offers are regulated by the T.U.F. ("Testo Unico della Finanza"), under the name **offerta pubblica di acquisto**, and they are defined as "every offer, invitation to offer or promotional message, in any form made, for the purchase or exchange of financial products and targeted at a number of investor above that is specified in …" [My translation].

In many European countries (for instance, Italy and U.K.) there is the obligation to issue a tender offer (so called **mandatory tender offer**) when an investor or a group of investors acquires more than a certain percentage of shares with voting rights; in Italy such threshold is 30%.

However the obligation to issue a tender offer doesn't exist in all countries. A remarkable exception is represented by the U.S.; for further information we refer to (Greene, 2006 – pag. 8-5).

1.2 Solutions of games in characteristic function form

A solution of a game in characteristic function form is a set of imputations, obtained through the cooperation of all players, on the basis of the characteristic function in such a way to assign a payoff to each player.

In literature there exist different types of solutions of a game in characteristic function form, although in order to guarantee the existence and the uniqueness of the solution one can use a particular solution: the value.

A **value** is a vector of the expected payoffs assigned to the players defined on the basis of a bargaining model or of some axioms. In literature there exist different values: the two most used are the Shapley value and the Normalized Banzhaf value. Notice that both are based on some axioms; for further information we refer to the textbooks previously indicated.

The Shapley value (Shapley, 1953) assigns a payoff to each player on the basis of the following bargaining model:

- the coalitions' formation is done through the addition of individual players until the global coalition is formed;
- a payoff equal to $v(S) - v(S\backslash\{i\})$ is assigned to every player i added to coalition $S\backslash\{i\}$;
- it is assumed that all the possible ways to form the global coalitions have the same probability.

The payoff assigned to each player by the Shapley value can be computed using the following formula, where s is the number of players that belong to coalition S:

$$\phi_i = \sum_{for\ all\ S \subseteq N} \frac{(s-1)!(n-s)!}{n!}[v(S) - v(S \setminus \{i\})]$$

Unlike the Shapley value, the Normalized Banzhaf value does not take into account the order in which the players are added to a coalition. In fact, while for the computation of the Shapley value all the possible permutations of players are considered, in the case of the Normalized Banzhaf value we consider only all the possible combinations.

The contribution of each player is the sum of the contributions that he provides to all coalitions; in formula:

$$c_i = \sum_{\substack{for\ all\ S \subseteq N \\ i \in S}} [v(S) - v(S \setminus \{i\})]$$

The payoff assigned to each player by the Normalized Banzhaf value can be computed using the following formula:

$$\beta_i = \frac{v(N)}{\sum_{h=1}^{n} c_h} \cdot c_i$$

All the solutions seen so far are valid for any game in characteristic function form. In the case of simple games, the values computed for these games are called **power indices**.

1.3 The Shapley-Shubik power index

The Shapley-Shubik power index (Shapley et al., 1954) can be considered as the Shapley value calculated for a simple game.

The bargaining model at the base of this index is similar to the one defined for the Shapley value; in fact:

- a coalition is formed by adding a player to a pre-existing coalition (that may be also the empty one);
- if the coalition obtained in such way is winning, then the last player is crucial;
- these operations have to be repeated until all the possible permutations of players have been considered.

It is possible to compute the Shapley-Shubik power index for the *i*-th player using the following formula, where d_j^i is the number of coalitions of cardinality *j* that the player *i* makes winning:

$$\phi_i = \sum_{j=0}^{n-1} \frac{j!(n-j-1)!}{n!} d_j^i$$

The following example helps to clarify the method of calculation of this index:

$N = \{1, 2, 3\}$

The characteristic function of the game *v* is defined as follows:

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

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

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

The computation of the Shapley-Shubik power index follows the bargaining model previously shown. In the first column of the following table we show all the sequences of coalitions that can be obtained by adding another player to an existing coalition. The empty coalition is represented by Ø. Taking a look to the table it is possible to notice that in the second column we report the coalition that, following the addition of a player, becomes winning, while in the third column we report the player that made that coalition winning. Finally, in the fourth column we show the losing coalition that becomes winning by adding the player i.

Sequences of coalition	Coalition S	Player	Coalition $S\backslash\{i\}$
Ø (1) (1, 2) (1, 2, 3)	(1, 2)	2	(1)
Ø (1) (1, 3) (1, 2, 3)	(1, 3)	3	(1)
Ø (2) (1, 2) (1, 2, 3)	(1, 2)	1	(2)
Ø (2) (2, 3) (1, 2, 3)	(1, 2, 3)	1	(2, 3)
Ø (3) (1, 3) (1, 2, 3)	(1, 3)	1	(3)
Ø (3) (2, 3) (1, 2, 3)	(1, 2, 3)	1	(2, 3)

Table 1. Necessary computations for the calculation of the Shapley-Shubik power index

Since it is assumed that all the sequences of coalitions that lead to the formation of the global coalition have the same probability, the power indices of the players are given by the ratio between the number of times in which the players are crucial and the number of permutations of the players. Thus, the power indices of the players 1, 2 and 3 are respectively 4/6, 1/6 and 1/6.

The same result could also be obtained using the formula presented above; in such case the power indices of the players are computed as follows:

$$\phi_1 = \frac{0!\cdot 2!}{3!}\cdot 0 + \frac{1!\cdot 1!}{3!}\cdot 2 + \frac{2!\cdot 0!}{3!}\cdot 1 = \frac{4}{6}$$

$$\phi_2 = \frac{0!\cdot 2!}{3!}\cdot 0 + \frac{1!\cdot 1!}{3!}\cdot 1 + \frac{2!\cdot 0!}{3!}\cdot 0 = \frac{1}{6}$$

$$\phi_3 = \frac{0!\cdot 2!}{3!}\cdot 0 + \frac{1!\cdot 1!}{3!}\cdot 1 + \frac{2!\cdot 0!}{3!}\cdot 0 = \frac{1}{6}$$

1.4 The Normalized Banzhaf power index

The Normalized Banzhaf power index can be considered as the Normalized Banzhaf value computed for a simple game.

The Normalized Banzhaf power index assigns to each player a share of power proportional to the number of coalitions for which is crucial. The sum of such shares is equal to 1.

Consider the formula for the computation of the Normalized Banzhaf value. Assume that v is a simple game and consider that the global coalition N is winning. The formula for the computation of the Normalized Banzhaf power index is the following:

$$\beta_i = \frac{c_i}{\sum_{h=1}^{n} c_h}$$

where c_i is the number of times in which the player i is crucial.

We consider the example introduced in the previous section in order to show how to compute the index.

In the first column of the following table we list the coalitions obtained by combining the players. The players that are crucial for a given coalition are reported in the other columns. Furthermore, in the subsequent rows of the table, there is a 1 if the player is crucial for the coalition S, a 0, if it is a losing coalition and a '-' if the player does not belong to the coalition.

Coalition S	Player		
	1	2	3
(1)	0	-	-
(2)	-	0	-
(3)	-	-	0
(1, 2)	1	1	0
(1, 3)	1	-	1

(2, 3)	-	0	0
(1, 2, 3)	1	0	0
	3	1	1

Table 2. Necessary computations for the calculation of the Normalized Banzhaf p.i.

Using the formula for the computation of the Normalized Banzhaf power index, the power indices of the players are:

$$\beta_1 = \frac{3}{5} \qquad \beta_2 = \frac{1}{5} \qquad \beta_3 = \frac{1}{5}$$

Finally it is necessary to point out that in addition to Banzhaf, other authors have formulated independently the same index: James S. Coleman, Lionel S. Penrose and, according to a particular interpretation, Martin Luther; see (Gambarelli et al., 1999), (Banzhaf, 1965), (Coleman, 1971) and (Penrose, 1946).

1.5 Multicameral Games

A weighted majority game can be used to represent an assembly; in fact, for the approval of a decision, it is necessary that the sum of the weights of the players that belong to the coalition supporting such decision is higher or equal to the majority quota.

In the case in which the decision process involves two or more assemblies it is necessary to define a rule that allows approving a decision on the basis of the deliberations of all assemblies. This situation can be described by a multicameral game.

Consider a collection of games in characteristic function form (N, v^1), ..., (N, v^m) defined over the same set of players $N = \{1, \ldots, n\}$: a **multicameral game** is a game resulting from the unification of the games (N, v^1), ..., (N, v^m) on the basis of a specific rule.

A coalition $S \subseteq N$ is winning in the unified game if and only if it is winning in all the games. Such rule is considered adequate for applications in both politics and finance.

From the characteristic function of the unified game it is possible to compute the power indices of the game. Such indices, in general, are different from those computed in the individual games; in fact, a player can be crucial for a coalition in an assembly, but not in the other.

1.6 Few notes about Graph Theory

In this section we present some definitions regarding Graph Theory.

A **network** (N, L) is a graph composed by nodes (belonging to the set N) and by edges (belonging to the set L). Given N, let be $L^N = \{\{i, j\} | \{i, j\} \subseteq N, i \neq j\}$.

Given a network (N, L), two nodes i and j belonging to such network are:

- **connected**, if there exist a sequence of nodes (x_0, ..., x_m) such that $x_0 = i$, $x_m = j$ and (x_i, x_{i+1})$\in L$ for all $i \in \{1, \ldots, m\text{-}1\}$;
- **directly connected**, if $\{i, j\} \in L$;

A **component** of a network is a set of nodes C such that two nodes i and j belong to C if and only if they are connected or directly connected.

A network (N, L) can be:

- **complete**, if $L = L^N$;
- **connected**, if it has just one component;
- **cycle-free**, if it does not exist a sequence of distinct nodes (i_1, ..., i_{m+1}), $m \geq 3$ such that $\{i_k, i_{k+1}\} \in L$ for all $i \in \{1, \ldots, m\}$, $i_1 = i_{m+1}$ and where i_1, ..., i_m are distinct nodes;
- **cycle-complete**, if for every cycle (i_1, ..., i_m, i_1), then all the nodes in the cycle are directly connected.

Consider the following graph composed by 6 nodes and 6 edges.

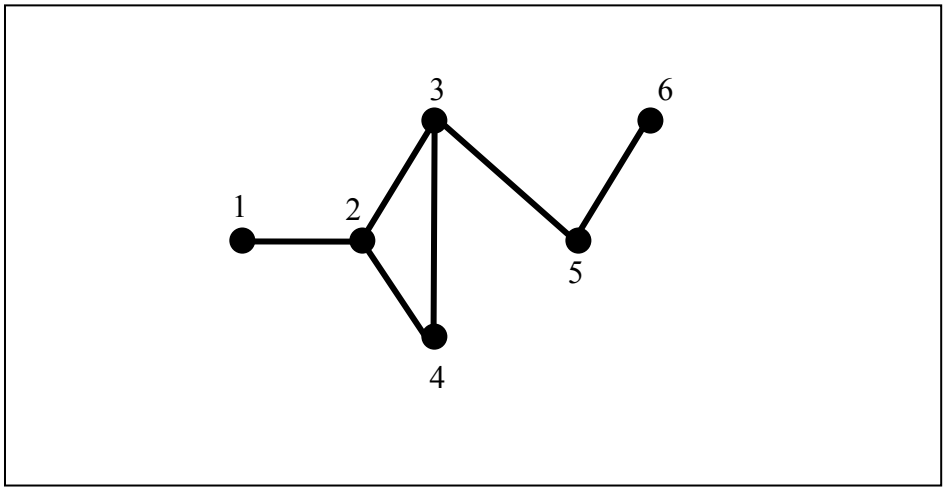


Figure 1. An example of graph

Notice that the network, having only one component, is connected. Furthermore, the network is cycle-complete; in fact, the players that belong to the unique cycle {2, 3, 4} are directly connected. However, the network is not complete.

1.7 A priori information about coalitions' formation

Up to this point it was assumed that the players could form any coalition. However in general one may have some a priori information that imposes restrictions to the cooperation among players making it difficult or impossible.

In literature there exist several models that allow taking into account this information (see for new developments (Khmelnitskaya, 2007)); the two most well-known are presented in the following subsections.

1.7.1 Coalition structures

The a priori information concerning the cooperation can be represented as a partition of the players according to the affinities between them. This approach, introduced and analyzed in (Aumann and Drèze, 1974) and (Owen, 1977), is defined as follows.

A **coalition structure** of the game is a partition Π of the set of the players in m coalitions (**a priori unions**), i.e.

$$\Pi = \left\{ N_i \subseteq N \middle| N_i \cap N_j = \varnothing, i \neq j, \bigcup_{i \in m} N_i = N \right\}$$

Given a partition Π, let be $M^{\Pi} = \{1, ..., m^{\Pi}\}$ the set of the elements of such partition.

A **game with coalition structure** (N, v^{Π}, Π) is composed by a game (N, v) and by a coalition structure (N, Π). For each game with coalition structure it is possible to define a **game among a priori unions** (M^{Π}, v^{Π}), where:

- M^{Π} is the set of a priori unions;
- $v^{\Pi}(Q) = v\left(\bigcup_{i \in Q} N_i\right) \qquad \forall Q \subseteq M^{\Pi}$

We refer to the articles mentioned above for an explanation of the methods used to adapt the different types of solutions to this type of games.

1.7.2 Communication situations

The a priori information concerning the cooperation can also be represented by a graph. Under this approach, introduced and analyzed in (Myerson, 1977), the a priori information is represented by a network (N, L).

A **communication situation** (N, v, L) is a triple composed by a game in characteristic function form (N, v) and by a network (N, L). Given a network and a coalition $S \subseteq N$ we define $S|L$ as a partition of players that belong to S who can coordinate their actions without the help of players outside S.

To such communication situation is associated a **network-restricted game** (N, v^L) where the characteristic function v^L is defined as follows:

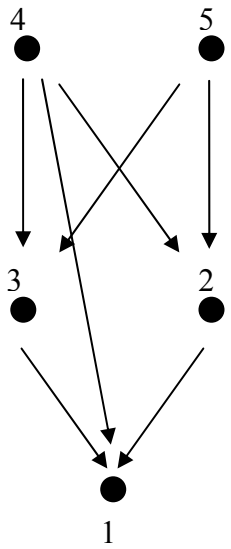
$$v^L(S) = \sum_{C \in S|L} v(C) \qquad \forall S \subseteq N$$

1.8 Reduction method for simple games

In this thesis we develop a new model for the analysis of corporate control. To do so it is necessary to present some methods, introduced in (Gambarelli and Owen, 1994), that allow identifying the investors that effectively control a particular company.

This method allows examining situation like the following one.

There are three companies 1, 2 and 3 and two investors 4 and 5. We assume that the investors are not controlled by any other player. The graph shown in Figure 2 represents the control structure, while the table contains the percentages of share owned by each player. Assume that a group of players in order to control a company must own the absolute majority of its shares.



	1	2	3	4	5
1	-	-	-	-	-
2	45	-	-	-	-
3	35	-	-	-	-
4	20	30	70	-	-
5	-	70	30	-	-

Figure 2. Example

Which coalitions of investors can control company 1? And which ones can control the other companies?

To answer to these questions it is necessary to present some definitions; see also (Gambarelli and Owen, 1994).

Let H be a finite set. A **clutter** over H is a collection W of subsets of H such that:

(a) $\varnothing \notin W$

(b) $H \in W$

(c) $S \in W \wedge S \subset T \subset H \Rightarrow T \in W$

A clutter can be:

- **proper**, if $\forall S, T \in W \qquad S \cap T \neq \varnothing$

- **strong**, if $\forall S \subset H \qquad S \in W \Leftrightarrow H \setminus S \notin W$

- **decisive**, if it is proper and strong

A **formal game system** (f.g.s.) for companies N and investors M is an n-tuple $W = [W_1, ..., W_n]$ of clutters over the set $N \cup M$.

A **reduction** for companies N and investors M an n-tuple $V = [V_1, ..., V_n]$ of clutters over the set M.

Let be W a clutter over H. A player $i \in H$ is called **dummy** for the clutter W if for every $S \in W$, both S and $S \setminus \{i\}$ belong to W.

It is assumed that every company i is dummy in the clutter W_i.

Returning to the previous example, notice that:

$N = \{1, 2, 3\}$

$M = \{4, 5\}$

The shareholder's meeting of company i is represented by the clutter W_i, thus:

$W_1 = \{(2, 3), (3, 4), (2, 4)$ and supersets$\}$

$W_2 = \{(5), (4, 5)\}$

$W_3 = \{(4), (4, 5)\}$

The formal game system W is thus composed by $[W_1, W_2, W_3]$.

Since there are no cross-ownerships between the two companies, it is possible to compute a reduction using the following algorithm:

1) Renumber the companies so that the company i is dummy for the clutters $W_j (i \leq j \leq n)$;

2) Let be $i = n$;

3) Compute the set of coalitions, composed only by companies, which are winning in the company i, i.e.

$$V_i = W_i \cap 2^N$$

4) For each $S \subset M$ let be $J_i(S)$ the set of the companies controlled by S, i.e.

$$J_i(S) = \left\{ J \middle| i+1 \leq j \leq n \; S \in V_j \right\}$$

5) Identify the coalitions of investors that control the company i, i.e.

$$V_i = \left\{ S \middle| S \subset M \wedge S \cup J_i(S) \in W_i \right\}$$

6) If $i > 1$, then decrease i by 1 and go back to step 3. Otherwise stop.

The n-tuple $V = [V_1, V_2, V_3]$ is defined the **effective reduction** of the f.g.s. W.

Here are listed the computational steps made by the algorithm in order to give the solution of the previous example:

Step 1: Since the companies are already numbered following the criteria, the renumbering is not necessary

Step 2: $i = 3$

Step 3: $V_3 = W_3 \cap \{(1,2),(1,3),(2,3),(1,2,3)\} = \varnothing$

Step 4: $J_3(\{4\}) = \varnothing$, $J_3(\{5\}) = \varnothing$, $J_3(\{(4,5)\}) = \varnothing$

Step 5: $V_3 = \{(4),(4,5)\}$

Step 6: $i = 2$

Step 3: $V_2 = W_2 \cap \{(1,2),(1,3),(2,3),(1,2,3)\} = \varnothing$

Step 4: $J_2(\{4\}) = 3$, $J_2(\{5\}) = \varnothing$, $J_2(\{(4,5)\}) = 3$

Step 5: $V_2 = \{(5),(4,5)\}$

Step 6: $i = 1$

Step 3: $V_1 = W_1 \cap \{(1,2),(1,3),(2,3),(1,2,3)\} = \{(2,3)\}$

Step 4: $J_1(\{4\}) = 3$, $J_1(\{5\}) = 2$, $J_1(\{(4,5)\}) = \{(2),(3)\}$

Step 5: $V_1 = \{(4),(4,5)\}$

Step 6: stop

Thus the effective reduction of W is composed by:

$V_3 = \{(4),(4,5)\}$

$V_2 = \{(5),(4,5)\}$

$V_1 = \{(4),(4,5)\}$

Notice that investor 4 alone controls both companies 1 and 3, while investor 5 alone controls company 2. The control of company 1 by investor 4 is possible through the control of company 3.

In the case in which there are cross-ownerships one has to resort to a more general solution method.

Consider the f.g.s. W and a reduction V. Then for each $T \subset M$ define

$$K(T) = \left\{ j \middle| T \in V_j \right\}$$

$$I(T) = K(T) \cup T$$

$$V_j^{'} = \left\{ T \middle| T \subset M \wedge I(T) \in W_j \right\}$$

It is possible to prove that V' is a reduction. Furthermore notice that the previous relationships define a mapping from a reduction to another reduction; let be Γ such mapping. A **consistent reduction** is a reduction V of f.g.s. W if it is a fixed point of the mapping Γ induced by W. It has been proved that any f.g.s. has at least one consistent reduction.

Moreover it can be proved that:

- if W is a proper f.g.s., then it has a proper reduction;
- if W is a strong f.g.s., then it has a strong reduction;
- if W is a decisive f.g.s., then it has a decisive reduction.

The following algorithm allows identifying the consistent reduction(s):

a) for each W_j (clutter over $N \cup M$) compute its multilinear extension, i.e.

$$MLE_j(x_1,...,x_{n+m}) = \sum_{S \in W_j} \left\{ \prod_{k \in S} x_k \prod_{k \notin S} (1 - x_k) \right\}$$

b) compute the solutions RE_j $(1 \leq j \leq n)$ of the system of equations

$$RE_j = MLE_j(RE_1,...,RE_n, x_{n+1},...,x_{n+m})$$

c) since that the solutions are ratios of multilinear functions that represent the reduced game, it is possible to identify all the controlling coalitions.

This algorithm can also be used to compute the efficient reduction. In order to show how to use the algorithm, consider the previous example.

For convenience of the reader, we report here some data regarding the previous example.

$N = \{1, 2, 3\}$

$M = \{4, 5\}$

$W_1 = \{(2, 3), (3, 4), (2, 4)$ and supersets$\}$

$W_2 = \{(5), (4, 5)\}$

$W_3 = \{(4), (4, 5)\}$

Step a: Compute the multilinear extension for every clutter W_j

$$MLE_1(x) = x_2x_3(1-x_4) + x_2x_4(1-x_3) + x_3x_4(1-x_2) + x_2x_3x_4$$

$$MLE_2(x) = x_5$$

$$MLE_3(x) = x_4$$

Step b: Solve the following system of equations

$$RE_1 = x_2x_3(1-x_4) + x_2x_4(1-x_3) + x_3x_4(1-x_2) + x_2x_3x_4 = 2x_4x_5(1-x_4) + x_4$$

$$RE_2 = x_5$$

$$RE_3 = x_4$$

Step c: Since $x_j \in \{0,1\}$, then $x_j = x_j^2$. Thus the reduced extensions are:

$$RE_1 = x_4 \qquad RE_2 = x_5 \qquad RE_3 = x_4$$

Thus, as before, the only crucial player for company 1 and 3 is investor 4, while the only crucial one for company 2 is investor 5.

In the third chapter it is shown how this method allows identifying the investors that control the different companies.

CHAPTER 2

My published results

In this chapter we summarize a model introduced in (Gambarelli and Uristani, 2009); such model, taking into account the cohesion of the players, extends the possible applications of the multicameral games.

2.1 Introductory example

Consider the following situation in which two assemblies vote on a certain subject; let be (N, v^1) the game related to the first assembly and (N, v^2) the game related to the second one.

Weights of game (N, v^1)		*Weights of game* (N, v^2)	
A (20%)	2	A (15%)	2
B (20%)	1	B (20%)	1
C (30%)	1	C (20%)	1
D (30%)	3	D (45%)	3

Table 3. Introductory example

In the table the players are divided into three groups on the basis of the different proposals. Assume that a player that belong to group 1 can not join a coalition to which belong a player that belongs to group 3 because their proposals are incompatible. In such case in the first assembly the only winning coalition is (A, B, C), while in the second one the winning coalitions are (A, D) and (A, B, C). Thus, in the unified game, the only winning coalition is (A, B, C). Since the player D does not belong to any winning coalition in the unified game, he has a power index equal to zero even if he has 30% of votes in game (N, v^1) and 45% of votes in game (N, v^2).

2.2 Model

Let be $N = \{1, \ldots, n\}$ the set of players that belong to the cooperative games $(N, v^1), \ldots, (N, v^m)$ in characteristic function form.

A **low unified game** of $(N, v^1), \ldots, (N, v^m)$ is a game $(N, v^{\downarrow})$ where the characteristic function is defined as follows:

$$v^{\downarrow}(S) = \min_{h \in \{1,\ldots,m\}} v^h(S)$$

To take into account the different probability of formation of the coalitions, to each coalition $S \subseteq N$ is assigned a **cohesion index** $c(S) \in [0, 1]$. If the cohesion index is equal to 1, then there is the maximum degree of cohesion among the players, while if such index is equal to 0, then the players cannot form that coalition.

For every $S \subseteq N$, $v^*(S)$ is obtained multiplying the value of the coalition in the low unified game for the related cohesion index. The game (N, v^*) is called **multicameral cohesion game**.

A multicameral cohesion game is called **multicameral simple cohesion game** if the games (N, v^1), …, (N, v^m) are simple games.

Finally, a multicameral cohesion game is called **multicameral weighted majority cohesion game** if the games (N, v^1), …, (N, v^m) are weighted majority games.

It should be noted that this model allows to describe situations with the following characteristics:

- a proposal must obtain the approval of each assembly;
- the decision of the majority binds all the players;
- each assembly is composed by the same n players.

Some applications of the model in finance and in economics have been presented in (Uristani, 2007); in particular, the model has been applied to the case of corporate mergers and of bankruptcy agreements.

In the case of a merger between two companies the decision to merge requires the approval of the shareholders' meeting of both companies. Moreover in each company the shareholders have some relationships among them based on their different interest in the plan of merger.

Another application is related to the case of financially distressed companies. In these cases it is possible for a company to obtain an debt restructuring agreement that allows to resume the business through a plan approved by the shareholders and creditors. Also in this case the interest of the players depends on different objectives: some may only be interested in the immediate recovery of their credit while others may be also interested in the continuation of the business.

CHAPTER 3

Further results in this thesis:
model for the representation of corporate control

In this chapter we present a new model for the representation of corporate control. Some applications of this model to Economics and Finance are introduced in the next section. After defining the mathematical model we present some remarkable situations that can be represented by this model.

3.1 Introduction and motivations

The model proposed here has been developed to support the analysis of problems in which it is necessary to identify the effective controllers of a given company. Such class of problems occurs in economics, in finance and in law.

In finance it is necessary to test the robustness of a control network against takeovers taking into account the relationships among the players. Furthermore, based on this model, it is possible to define takeover strategies; in this case the acquisition of the control of the target company can be done through the purchase of shares that belong to other companies that are associated with it. This is particularly beneficial for at least three reasons: first the price of the shares of the other companies can be lower than those of the target company, second the massive purchase of the shares of a company leads to large fluctuations of the price, and finally in such way the raider is not excessively exposed towards the target company since he diversifies his investment.

Furthermore there are important applications to Corporate Finance. An issue frequently analyzed in literature is the identification of the value of shareholders' voting rights. A possible approach, used in (Nenova, 2003), considers the normalized Shapley value as a measure of the decision power of a shareholder. However the computation of such value on the basis of the direct shareholders of a company does not allow to capture the effect of voting agreements and cross-ownerships. Through the application of the proposed model it is possible to measure accurately the decision power of each shareholder.

In economics, the Anti-Trust authorities need to verify if there are groups that control a set of companies operating in the same economic sector. In this case the existence of such groups can lead to the manipulation of the economic activities. The model, identifying such control groups, provides support to the investigations conducted by the Anti-Trust authorities.

Moreover, in different countries, the authorities for the control of the regulated markets need to know which investors have effectively a strong influence in a company.

Furthermore it exist the necessity to compute the amount of shares of a company owned by a big investor. In such case one has to consider all the indirect participations, since some rules (like those relative to mandatory tender offers) apply to an investor if he owns a percentage of shares that exceeds a certain threshold (f.i., 30% in Italy).

In literature a method that takes into account the different aspects of indirect control among companies has been introduced in (Gambarelli and Owen, 1994). Furthermore in (Denti e Prati, 2001) has been presented an algorithm for the computation of the winning coalitions that can be used for applying the method mentioned above.

The goal of this first contribution is the development of a method that allows to identify the individuals that effectively control the decisions in the assembly of every company. In this case there are many problems:

- the type and the number of majorities required to gain control of a company (f.i. in case of so-called "Società Cooperative" or in case of golden shares) other than the simple majority;
- the existence of voting agreements among the investors.

To consider these problems it is necessary to develop a new model based on the multicameral voting games introduced in (Gambarelli and Uristani, 2009).

The work presented in this chapter consists of two parts: the first on the determination of the set of winning coalitions, the second on the identification of those who control the different companies.

Regarding the first part, the model allows to represent the different factors that define which coalitions are winning and which are losing. In particular, two aspects are considered: the connections among groups of investors and the different voting systems. In fact, there exist different voting systems according to the type of company: for example, in the Italian "Società Cooperative" (cooperative companies) each member of the company has one vote regardless of the quotas of the company that he owns. In addition the voting behaviors can be influenced by voting agreements.

Regarding the second part, on the basis of the results provided by the model it is possible to use the method introduced in (Gambarelli and Owen, 1994) to identify the investors that effectively control the different companies.

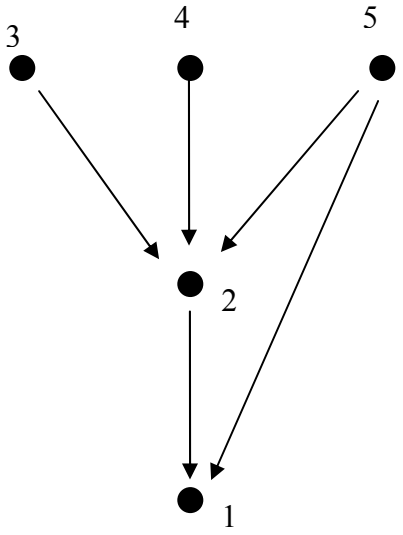
The model, given the shares owned by each player in each company, the voting agreements among investors and the procedures for the approval of the decisions in each company, allows to identify the coalitions that effectively control a company.

Remark. In this thesis it is assumed that the investors that join a voting agreement in a certain company commit themselves with all their shares.

3.2 Model description

The model is introduced by means of an example.

There are two companies (1 and 2) and three investors (3, 4 and 5). The first company is controlled by investor 5 and by company 2. The latter is controlled by all investitors. A graphical representation of this situation is given by the following graph; the following table shows the shares owned by the players in the two companies.



	1	2
1	-	-
2	55	-
3	-	35
4	-	30
5	45	35

Figure 3. Description of the example

Furthermore, we assume that in the second company exists a voting agreement between investors 3 and 4. The relationships among the investors are represented as follows:

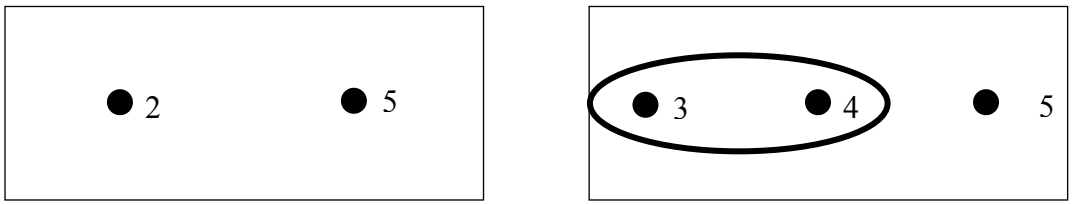


Figure 4. Graphical representation of the relationships

This example is formalized as follows.

Let be I and N respectively the set of investors and the set of companies. Let be P the number of players.

For each company $k \in N$ define:

- h_k : number of decision conditions;
- w_k : matrix (P x h_k) of weights of the players;
- q_k : vector (h_k x 1) of majority quotas (>50%);
- Q_k : minimum number (>0) of decision conditions that has to be respected.

Remark. In this thesis we assume that a company does not own its shares. Notice that this assumption is not too restrictive since in many countries a company cannot exercise the voting rights for those shares. In fact, in that case the managers of the company can influence or control the company itself. Finally, we could not find a country that adopts different rules in this regard.

To represent the decision structure of each company a new type of multicameral game is developed.

For each $k \in N$, define a multicameral game $G_k = (P_k, v_k)$ where:

- the set of players is $P_k = \left\{ i \in I \cup N \middle| w_k^h(i) > 0 \text{ for at least an } h \right\}$;
- the characteristic function is $v_k(S) = \chi\left(\sum_h \chi\left(w_k^h(S) \geq q_k^h \right) \geq Q_k \right) \quad \forall S \subseteq P_k$.

The relationships among the players are represented through a coalition structure (P_k, Π_k). Given a partition of P_k, $\Pi_k = \left\{ N_i \subseteq N \middle| N_i \cap N_j = \varnothing, i \neq j, \bigcup_{i \in M_k} N_i = N \right\}$, define $M_k = \{1, ..., m_k\}$ the set of the elements of this partition. Two or more players that sign a voting agreement belong to the same a priori union.

Remark. It is assumed that no company is allowed to join a voting agreement.

For each $k \in N$, consider a multicameral game with coalition structure (P_k, v_k, Π_k), composed by the game G_k and by the coalition structure (P_k, Π_k). Define a game among a priori unions C_k, where:

- the set of players is M_k;

- the characteristic function is $v_k^{\Pi}(Q) = v_k\left(\bigcup_{i\in Q} N_i\right)$ $\forall Q \subseteq M_k$

In this thesis these games are called **company games**.

In this way we defined the model used for representing the decisions of each company. Now we show how to utilize this model to identify the investors that effectively control each company. Thus now it is necessary to present some propositions and a lemma.

Proposition 1. Each company game C_k is a simple game.

Proof.

$$v_k^{\Pi}(Q) = v_k\left(\bigcup_{i\in Q} N_i\right) = \chi\left(\sum_h \chi\left(w_k^h\left(\bigcup_{i\in Q} N_i\right) \geq q_k^h\right) \geq Q_k\right) \qquad \forall Q \subseteq M_k$$

Since $v_k(S)$ can be equal to 0 or 1 for any coalition S, $v_k^{\Pi}(Q)$ must be equal to 0 or 1 for any coalition Q.

■

It is well-known that, for each game C_k, the set of **winning coalitions** is defined as follows:

$$W_k = \left\{\bigcup_{i\in Q} N_i \subseteq P_k \middle| v_k^{\Pi}(Q) = 1\right\}.$$

Lemma 1. W_k is a clutter.

Proof.

(a) $\varnothing \notin W_k$

Since $v_k^{\Pi}(\varnothing) = 0$, then this coalition does not belong to W_k

(b) $M_k \in W_k$

$$v_k^{\Pi}(M_k) = v_k\left(\bigcup_{i\in M_k} N_i\right) = \chi\left(\sum_h \chi\left(w_k^h(P_k) \geq q_k^h\right) \geq Q_k\right) =$$

$$= \chi\left(\sum_h \chi\left(1 \geq q_k^h\right) \geq Q_k\right) = \chi(h \geq Q_k) = 1$$

Thus, by definition of W_k, $M_k \in W_k$.

(c) $S \in W_k \wedge S \subset T \subset M_k \Rightarrow T \in W_k$

$$v_k^{\Pi}(T) = v_k\left(\bigcup_{i\in T} N_i\right) = \chi\left(\sum_h \chi\left(w_k^h\left(\bigcup_{i\in T} N_i\right) \geq q_k^h\right) \geq Q_k\right) =$$

$$= \chi\left(\sum_h \chi\left(w_k^h(\bigcup_{i\in S} N_i) + w_k^h(\bigcup_{i\in T/S} N_i) \geq q_k^h\right) \geq Q_k\right)$$

Since $v_k^{\Pi}(S) = 1$ and $w_k^h(\bigcup_{i\in T/S} N_i) \geq 0$, then $v_k^{\Pi}(T) = 1$

Thus, by definition of W_k, $T \in W_k$.

■

Proposition 2. $W = [W_1, ..., W_n]$ is a formal game system over $I \cup N$.

Proof. By Lemma 1, each W_k is a clutter. Thus W is a n-tuple of clutters, i.e. a formal game system.

■

Notice that, on the basis of Proposition 2 and of Lemma 1, the model here presented leads to the calculation of the related formal game system. Thus in order to find the investors that effectively controls each company it is possible to use the method presented in section 1.8.

Now it is possible to compute the solution of the previous example.

For convenience of the reader, here are reported some data of the example.

$N = \{1, 2\}$

$I = \{3, 4, 5\}$

First, for each company game C_k, we identify the partitions of the players:

$\Pi_1 = \{(2), (5)\}$

$\Pi_2 = \{(3, 4), (5)\}$

Thus, for each company game C_k, we identify the winning coalitions:

$W_1 = \{(2)$ and supersets$\}$

$W_2 = \{(3, 4)$ and supersets$\}$

Then we compute the multilinear extensions of the clutters W_1 and W_2.

$$MLE_1(x) = x_2$$

$$MLE_2(x) = x_3 x_4$$

Then we solve the following system of equations:

$$\begin{cases} RE_1 = x_2 \\ RE_2 = x_3 x_4 \end{cases}$$

Finally, the effective reduction of this formal game system is the following:

$$RE_1 = x_3 x_4 \quad RE_2 = x_3 x_4$$

Thus investor 3 and 4 control both companies; this is due to the voting agreement in company 2 between investors 3 and 4.

To highlight how important is to take into account the voting agreements, consider the case described in the previous example, but ignoring the voting agreement.

In this case, for each company game C_k, we identify the set of winning coalitions:

$W_1 = \{(2)$ and supersets$\}$

$W_2 = \{(3, 4), (3, 5), (4, 5)$ and supersets$\}$

The effective reduction of this formal game system is the following:

$$RE_1 = x_3 x_4 + x_3 x_5 + x_4 x_5 - 2x_3 x_4 x_5$$

$$RE_2 = x_3 x_4 + x_3 x_5 + x_4 x_5 - 2x_3 x_4 x_5$$

Notice that in this case the coalition (3, 4) is not the only coalition that can control both companies.

3.3 A remark about games with graph-restricted communication

The reader may wonder why we did not use the network-restricted games. In this section we explain the reasons of this choice.

For each $k \in N$, define a communication situation (P_k, $v_k^{L_k}$, L_k) composed by the game G_k and by the network (P_k, L_k). The network-restricted game NRG_k is composed by:

- the set of players P_k;
- the characteristic function $v_k^{L_k}(S) = \sum_{C \in S|L_k} v_k(C) \qquad \forall S \subseteq P_k$.

It is possible to notice that a game NRG_k may not be a simple game. In fact consider the following situation with three players and with $Q_k = 1$ and $q_k = 50.01\%$:

	1	2	3
decision condition 1	60	20	20
decision condition 2	20	60	20
decision condition 3	20	20	60

Table 4. Weights of the players

The characteristic function is the following:

$$v_k^{L_k}(S) = \sum_{C \in S|L_k} v_k(C) = \sum_{C \in S|L_k} \chi\left(\sum_h \chi\left(w_k^h(C) \geq q_k^h\right) \geq Q_k \right) \qquad \forall S \subseteq P_k$$

Assume that the network has only one link between 1 and 2. Consider the coalition (1, 2, 3); in this case the value of the coalition is 2. Thus the game is not simple.

Thus in this case is not possible to use this representation with the method presented in section 1.8.

However, even if the network-restricted games are simple, the main problem is given by the fact that in the case of network-restricted games it may happen that some coalitions are considered winning while they are not even feasible.

Consider the following example. There are three players and a voting agreement between investors 1 and 2. To the left the situation is represented by a graph, while to the right it is represented by a partition of the set of players.

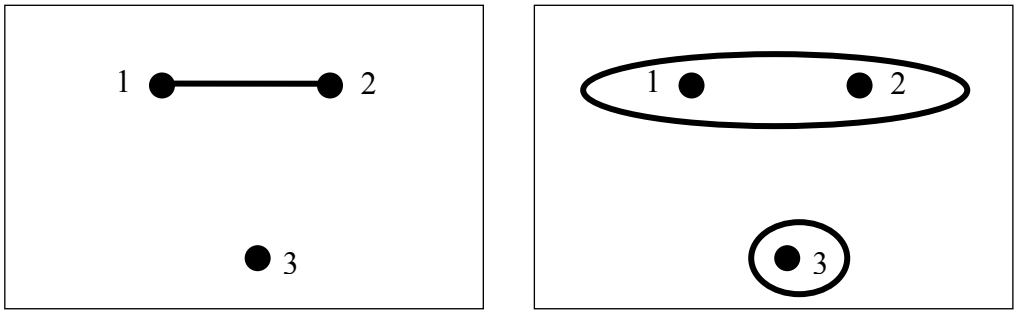


Figure 5. Relationships among the investors

Consider the coalition (1, 3).

In the first case $v_k^{L_k}((1,3)) = \sum_{C \in S|L_k} v(C) = v(1) + v(3)$. If the coalition (1) or the coalition (3) is winning, then also the coalition (1, 3) is winning. But this is not correct since that the coalition (1, 3) is not even feasible since it violates the voting agreement (investor 1 cannot join a coalition without investor 2).

In the second case M_k = (1, 2) where the first element of M_k represents the set of players 1 and 2, while the second element is relative to player 3. Thus in this case it is not possible that the formation of coalition (1, 3). Moreover, all the coalitions that can be formed in a game with a priori unions are feasible.

3.4 Dealing with the float

In many countries the laws prescribe that who owns a certain percentage of shares with voting rights that exceeds a specific threshold is obliged to give such information to the public. However, it may happen that part of the shares of a company is owned by small investors; these investors own only a small percentage of shares and thus they are not obliged to make a public declaration. As a consequence it is not know who these investors are.

The float F_k is the fraction of quotas of ownership that has voting rights in the company k for which we do not know the owner.

There are different ways to deal with the float. Some of these are surveyed in (Crama and Leruth, 2007); in particular they present the following situations:

- the case in which the float is divided in equal parts to some fictionary players; the percentage of share assigned to each player is equal to the one owned by the smallest known shareholder;
- the case in which the number of shareholders is infinite and each player owns a vanishingly small fraction of shares. Some results have been provided in literature for the computation of some power indices;
- the case in which the float is modeled using a random variable.

In this thesis we deal with the float in a different way. The reason is that we consider that the float is owned by investors that are interested in the investment itself, rather than in the control of a company. This is motivated by the consideration that otherwise, given the advantages resulting from the control (including the possibility to extract private benefits), they would try to have a share that is enough to influence the decisions. Thus the float is not represented by any player. However, this imposes a restriction on the model that thus is applicable only to situations in which there is at least a winning coalition.

This approach will turn out to be useful in the analysis presented in later chapters.

3.5 Some remarkable cases

In this section some important cases are shown that can be represented by this method. Unless otherwise specified, we are referring to cases arising under Italian law. However such kind of regulations is also enforced in other countries.

A first interesting case is the case of the “Società Cooperative” (cooperative companies). The Italian law defines such type of company in the art. 2511 of the “Codice Civile”. In such company the voting rights do not depend on the amount of quotas owned by an individual, but it is rather regulated by a one head-one vote system. Furthermore, recent legislation (art. 2540 c.c.) introduced the possibility, and in some cases the obligation, to divide the members of the company in separated assemblies (f.i. on a geographic base). In this assemblies are elected delegates who will vote at the assembly of the delegates.

This case can be represented as follows:

- P_k is the set of the players;
- h_k is the number of assemblies (excluding the assembly of the delegates);
- w_k is the matrix of the weights

$$w_k(i,j) = \begin{cases} \dfrac{1}{number\ of\ \ players\ in\ assembly\ i} & for\ all\ \ players\ \ j \in assembly\ i \\ 0 & for\ all\ \ players\ \ j \notin assembly\ i \end{cases}$$

- q_k is the majority quotas in the assemblies;
- Q_k is the minimum number of delegates that represent the majority in the assembly of the delegates (equal to majority quota of the assembly of delegates multiplied by h_k).

With these data it is possible to use the model presented in this chapter.

Another interesting case is represented by a situation in which there exist golden shares. A golden share, typically held by sovereign states, is a share that gives to the holder the right to be crucial in the decision process for some kind of decisions. Also in this case the model can be used to analyze the situation in the case of those specific decisions. Let be A the number of individuals who owns a golden share; the situation can be represented as follows:

- P_k is the set of players;
- h_k is equal to two;
- w_k is the matrix of the weights (the first row contains the percentages of ownership of each player, while the second row contains the elements equal to $1/A$ in case of players who own a golden share and equal to 0 otherwise);
- q_k is a vector of two elements: the first element is equal to the majority quota in the assembly, while the second one is equal to 1;
- Q_k is equal to two.

Also this situation can be represented by the model introduced in this chapter.

In this chapter we have shown a model for the representation of corporate control and some comments related to the possible cases that can be analyzed with it.

CHAPTER 4

Further results in this thesis:

model for the representation of corporate control: Algorithms

In this chapter we present some algorithms for the identification of the control groups. The structure of the input data is the same for all the algorithms, even if the output is different. In fact, each algorithm allows achieving different goals. We have included the Matlab code since we think that this language can be easily understood and since we provided some comments. In this way it is possible to give an accurate description of the algorithms.

4.1 The structure of input data

In this section is reported the structure of the input data. The classification of the variables is based on the Matlab language.

The set of companies (numbered from 1) is represented by vector *N*, while the set of investors (numbered from number of companies plus 1) is represented by vector *I*.

The percentages of ownership of the players are represented by variable *w*.

The majority quotas and the number of minimum conditions to be satisfied are represented respectively by the variables *q* and *Q*.

Finally, the a priori unions are represented by the variabile *Apriori*. For each *k*, every column of the vector *Apriori*{*k*} represents a player. Every group of players that has joined a voting agreement is represented by the same number.

For example, here is a possible instance of the data structure.

```
I = [3 4 5];
N = [1 2];

w{1} = [0 20 10 20 35; 0 10 10 40 20];
w{2} = [20 0 10 30 25];

q{1} = [50.01; 59.01];
q{2} = [50.01];

Q{1} = 2;
Q{2} = 1;

Apriori{1} =[1 1 2];        % In this case players 3 and 4 have signed
                            % a voting agreement
Apriori{2} =[1 2 3];
```

Notice that such data can be found in reality. For example in the case of the Italian regulated markets, the information about voting agreements and about ownerships that are greater than 2% are published on the website of the authority that regulates the markets, the CONSOB.

4.2 An algorithm for the computation of the reduced extensions

This first algorithm applies to any type of situation and allows identifying the reduced extension of each game. The algorithm applies the method developed in the previous chapter by using symbolic computation.

The algorithm is illustrated by the following flow-chart.

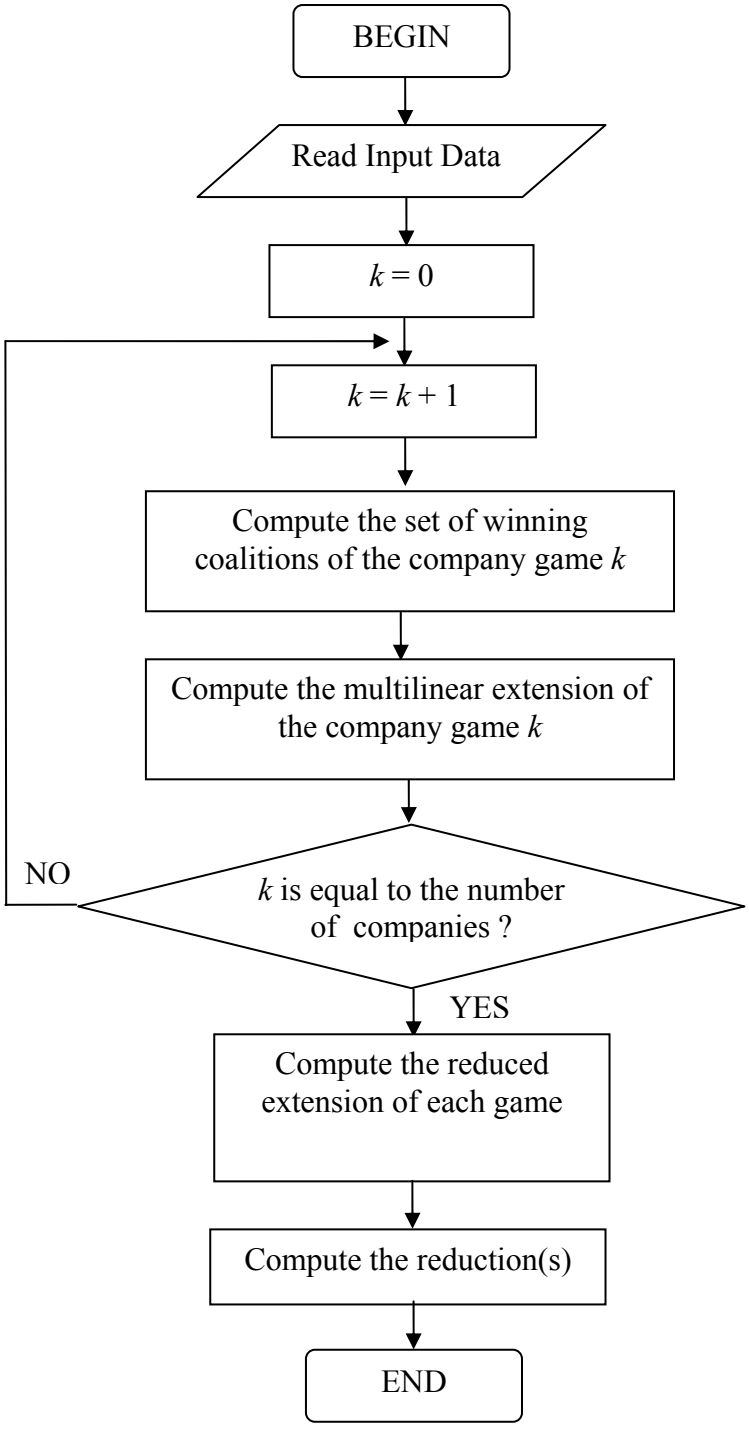


Figure 6. Flow-chart representation of the algorithm

To illustrate in details the operations carried out by the algorithm, we show the Matlab code with the related comments. Notice that in the area where we wrote the comment "INSERT INPUT HERE" one must include the input data.

```
% INSERT INPUT HERE
% For all companies
```

```
for k=[1:1:length(N)]

    % Computes the number of decision conditions of the company game k
    h{k} = length(q{k});

    % Finds the sets of players (companies and investors) that has some
    % participations in the company k
    PN{k} = N(find(sum(w{k}(:,N),1)>0));
    PI{k} = I(find(sum(w{k}(:,I),1)>0));

    % Computes the company game k
    [G_quot] = compute_company_game(PN{k}, PI{k}, N, I, w{k}, q{k}, Q{k}, ...
                                                                Apriori{k});

    % Finds the set of the winning coalition
    cont = 0;
    W = {};
    for u = [1:1:length(G_quot)]
        if G_quot(u).v
            cont = cont + 1;
            W{cont} = G_quot(u).Global_S;
        end
    end

    % Computes the multilinear extension for every company
    mult = [];
    for i = [1:1:length(W)]
        other = setdiff(union(PI{k},PN{k}),W{i});
        if length(other)>0
            temp = sprintf('(1-x%d)*', other);
            mult = [mult  sprintf('x%d*', W{i}) temp];
            mult = [mult(1:1:length(mult)-1) '+'];
        else
            mult = [mult  sprintf('x%d*', W{i})];
            mult = [mult(1:1:length(mult)-1) '+'];
        end
    end

    mult_lin{k} = [mult(1:1:length(mult)-1)];

end

% Creates the system of equations to be solved
for k = [1:1:length(N)]
    f{k} = [mult_lin{k} '-x' num2str(k)];
    f{k} = simple(sym(f{k}));
    f{k} = horner(f{k});
    inc{k} = ['x' num2str(k)];
end
```

The algorithm here presented allows formulating a system of equations that has to be solved to obtain the reduced extension of each company. To solve this system it is possible to use the function `solve`.

Notice that the code presented above calls a function for the computation of the company game. The input of such function is composed by:

- *PN*: the set of companies that owns some shares of the company;
- *PI*: the set of investors that owns some shares of the company;
- *N*: the set of all the companies;
- *I*: the set of all investors;
- *w*: the weights of the players in the company;
- *q*: the set of majority quotas;
- *Q*: the number of decision conditions;
- *Apriori*: the a priori unions.

The function computes the company game. In particular, it returns the characteristic function of such game; furthermore, for each coalition $S \subseteq M_k$ of the company game it returns also the coalition $\bigcup_{i \in S} N_i$.

```
function [G_quot] = compute_company_game(PN, PI, N, I, w, q, Q, Apriori)

    % Computes the investors of the company game
    P_quot = unique(Apriori(PI-length(N)));

    % Computes the weights of the investors
    w_prot = [];
    for j = [1:1:length(P_quot)]
        w_prot(:,j+length(PN)) = sum(w(:,PI(find(Apriori==j))),2);
    end

    % Computes the companies that have a participation of the company game
    P_quot = [PN P_quot+max(PN)];

    % Computes the weights of the companies
    for j = [1:1:length(PN)]
        w_prot(:,j) = sum(w(:,PN(j)),2);
    end

    cont = 0;
    for j = [1:1:length(P_quot)]
        % Computes the coalitions of cardinality j
        temp = nchoosek([1:1:length(P_quot)],j);

        for u = [1:1:size(temp,1)]
           cont = cont + 1;
           G_quot(cont).v = (sum(sum(w_prot(:,temp(u,:)),2)>=q)>=Q);
           G_quot(cont).S = temp(u,:);
           temp2 = [];
           for x = [1:1:length(temp(u,:))]
               if P_quot(temp(u,x))<=max(PN)
                   temp2 = [temp2 P_quot(temp(u,x))];
               else
```

```
                    temp2 = [temp2 ...
                             I(find(Apriori==(P_quot(temp(u,x))-max(PN))))];
                end
            end
            G_quot(cont).Global_S = sort(temp2,'ascend');
        end
    end
end
```

Once obtained the reduced extension for each company, it is possible to compute the related reduction(s).

A strong limitation of the use of this approach is the extreme slowness of the computation of the reduced extensions. This may become a problem as the number of players increases since in that case the computational time may be very high. However the advantage of this algorithm is the possibility, through such reduced extensions, to compute all the possible reductions. In particular, the importance of computing all the possible reductions is evident from the following example from (Gambarelli and Owen, 1994).

There are three companies and two investors. Every company owns 30% of the shares of the other companies. Every investor owns 20% of each company and there are no voting agreements. Notice that the coalition composed by all investors is a losing coalition in each company.

There exist three reductions for this formal game system:

- $V = (V_1, V_2, V_3)$ $\quad V_i = \{(4), (4, 5)\}$ for all $i = 1, 2, 3$
- $U = (U_1, U_2, U_3)$ $\quad U_i = \{(5), (4, 5)\}$ for all $i = 1, 2, 3$
- $Z = (Z_1, Z_2, Z_3)$ $\quad Z_i = \{(4), (5), (4, 5)\}$ for all $i = 1, 2, 3$

Consider for example the reduction V; in this case if the investor 4 is able to put in every company a loyal management, then he can keep the control of all the companies.

Notice that with the algorithm just presented it is possible to compute the list of coalition that can keep a stable control of the companies even if the coalition composed by all investors has not the majority of shares in all the companies. However we believe that the interpretation given to such reductions is not too appropriate for the analysis. First the companies should act only accordingly to the decisions of the shareholders' meeting, while in this case the management may directly influence the decisions of the assembly. Furthermore a goal of this thesis is to develop a model based only on available data: the information on the relationships between the management and the investors is not, in general, public or easy to be retrieved.

4.3 An algorithm for the general case

In this section we present an algorithm that determines, for each company, the set of the coalitions of investors that control such company. This algorithm is faster than the previous one, but it does not allow computing all the reductions. However it is important to notice that this algorithm also considers the cross-ownerships among the companies. From this section to the end of the thesis it is assumed that in each game C_k there exists at least a coalition S of investors such that $v_k(S) = 1$.

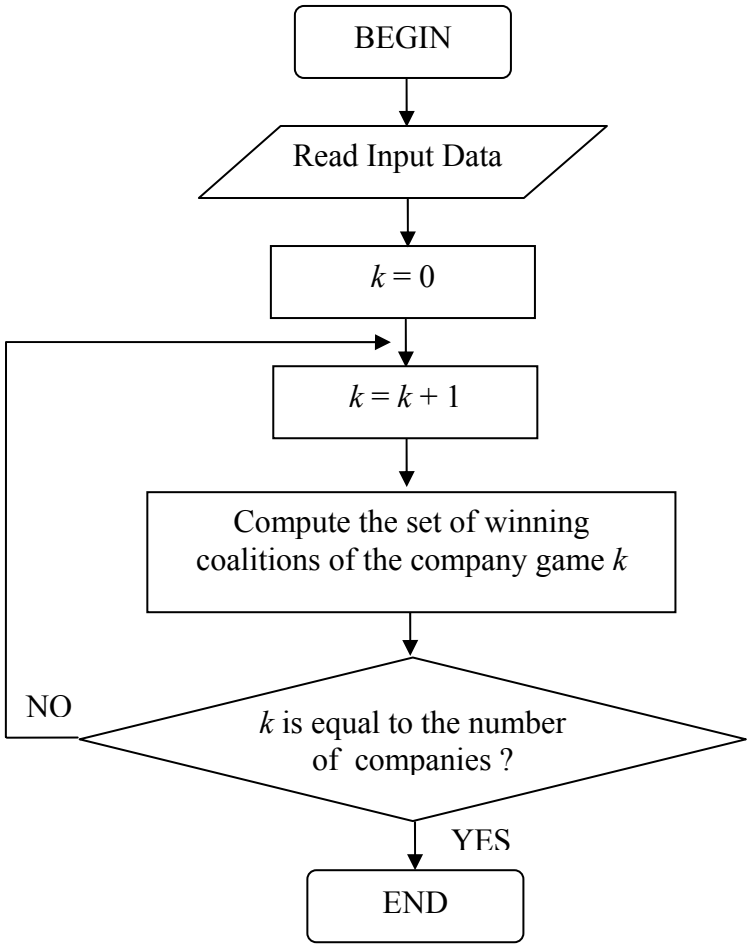


Figure 7. Flow-chart representation of the algorithm

Here is the code of the main algorithm. Such algorithm calls the function *compute_goodlist* which is described later.

```
% INSERT INPUT HERE

% Initializations
NW = [];
q_v = [];
Q_v = [];
NW_struct = [];
```

```
fcont = 0;
lcont = 1;

for u = [1:1:size(w,2)]
    NW = [NW; w{u}];
    fcont = size(w{u},1);
    q_v = [q_v; q{u}];
    Q_v = [Q_v; Q{u}];

    % NW_struct{u} indicates which rows of NW are related to company u
    NW_struct{u} = [lcont:1:lcont+fcont-1];
    lcont = lcont + fcont;
end

% Computes the winning coalitions for all companies
for target_firm = [1:1:length(N)]
    winn{target_firm} = compute_goodlist(N, I, NW, Apriori, NW_struct,  ...
                                                    q_v, Q_v, target_firm);
end
```

The function *compute_goodlist* is partly inspired by the algorithm presented in (Denti and Prati, 2001). This function receives the following input data:

- *N*: the set of the companies;
- *I*: the set of the investors;
- *NW*: the matrix of the weights of the players in the different companies;
- *NW_struct*: it indicates the rows of *NW* that are related to every company;
- *Apriori*: a priori unions;
- *q_v*: the set of the majority quotas;
- *Q_v*: the number of decision conditions;
- *target_firm*: the company for which compute the winning coalitions.

The function returns the set of the winning coalitions.

```
function [goodlist] = compute_goodlist (N, I, NW, Apriori, ...
                                        NW_struct, q_v, Q_v, target_firm);
% This function computes the set of winning coalitions for the target_firm.

% Computes the number of players
TOT = length(N) + length(I);

% Initializations
cont = 0;
count_good = 0;
goodlist = {};
exit_fl = 0;

h = waitbar(0,sprintf('Evaluating company game %d ...',target_firm));
for j = [1:1:length(I)]
```

```
    waitbar(j/length(I), h);

    % Computes the possible coalitions of cardinality j
    temp = nchoosek(I([1:1:length(I)]),j);

    fff = 0;
    tmp_goodlist = {};
    tmp_count_good = count_good;

    % Adds to the set of winning coalitions those that contains a winning
    % coalition that has been already found and that respect a priori unions
    % conditions for all companies
    for y = [1:1:tmp_count_good]
        fff = 1;
        for ds = [1:1:size(goodlist{y}, 2)]
            [ttt, dum] = find(temp==goodlist{y}(ds));
            if fff == 1
                fff = 0;
                uuu = ttt;
            else
                uuu = intersect(uuu, ttt);
            end
        end
        if not(isempty(uuu))
            for du = [1:1:size(uuu,2)]
                [Z_A] = A_Priori_Check(temp(uuu(du,:),:), N, Apriori);
                if Z_A(target_firm) == 1
                    count_good = count_good+1;
                    goodlist{count_good} = temp(uuu(du,:),:);
                    temp(uuu(du,:),:)=[];
                end
            end
        end
    end

    for u = [1:1:size(temp,1)]
        cont = cont+1;
        l = zeros(length(I),1)';
        ris = 1;

        % Computes the companies that are controlled by the coalition
        % temp(u,:)
        [Z] = compute_controlled_firms(temp(u,:), N, I, NW, NW_struct, ...
                                                                q_v, Q_v);

        [Z_A] = A_Priori_Check(temp(u,:), N, Apriori);

        new_Z = min(Z, Z_A);
        if new_Z(target_firm) == 0
            while ris
                old_Z = min(new_Z, Z_A);
                [nnZ] = compute_controlled_firms([find(old_Z.*N'>0)' ...
                                temp(u,:)], N, I, NW, NW_struct, q_v, Q_v);
                new_Z = nnZ;
                new_Z = min(new_Z, Z_A);
                ris = not(all(old_Z == new_Z));
            end
        end

        % If target_firm is controlled by the coalition, then add the
        % coalition in goodlist
```

```
        if new_Z(target_firm)>0
           count_good = count_good + 1;
           goodlist{count_good} = temp(u,:);
        end

    end

end

close(h);
```

The function *compute_goodlist* calls two functions: *compute_controlled_firms* and *A_Priori_Check*.

The function *compute_controlled_firms* allows to identify the companies directly controlled by a coalition given in input (called *coal*).

```
function [Z] = compute_controlled_firms (coal, N, I, matr_w, NW_struct, ...
                                                                    q_v, Q_v)

  % ZZ is the vector of all assemblies of all the companies
  % ZZ(i)=1 if the coalition wins in that assembly
  ZZ = sum(matr_w(:,coal),2)>=q_v;

  % initialization
  Z = zeros(length(N),1);

  % for all the companies
  for h = [1:1:length(N)]
      % Z is the vector of all companies that are controlled by the coalition
      Z(h) = (sum(ZZ(NW_struct{h}))>=Q_v(h));
  end
end
```

The function *A_Priori_Check* allows determining if a certain coalition is feasible given the information about the a priori unions. This function receives the following input data:

- *coal*: coalition to be analyzed;
- *N*: the set of the companies;
- *Apriori*: the informations about the a priori unions.

```
function [Z_A]=A_Priori_Check(coal, N, APriori)

    % initializations
    v = APriori;
    tmp_v = v;
    Z_A = ones(length(N),1);
```

```
    % for each company k
    for k = [1:1:length(N)]

       % Finds the vector of A Priori Unions for coalition coal
       coal_unions = v(k,coal-length(N));
       tmp_v(k,coal-length(N)) = -1;
       for j = [1:1:length(coal)]

            if find(tmp_v(k,:) == coal_unions(j))>0
                % If there is some player who is linked to one of the players
                % that belongs to coal doesn't belong to coal, then coal is not
                % feasible
               Z_A(k) = 0;
               break;
            end
       end
    end
end
```

The computational time of this algorithm is lower than the one of the previous algorithm. However for problems with a considerable number of players the time required for the computation is still high.

For example, consider the situation in which there are 10 companies and 15 investors and where for each company there exist from 5 to 10 players. There is no voting agreement. In this case the algorithm took about 15 minutes to solve the problem (with a laptop with 2 CPU 1.66 Ghz processor and 1 Gb of RAM).

4.4 A faster algorithm for pyramidal structures

In this section we present an algorithm suited for the cases in which the control structure is pyramidal; thus are excluded the cross-ownerships. These control structures are widely used for various reasons including, for example, the fact that in case of cross-ownerships the exercise of the voting rights is subject to restrictions by legal regulations.

Exploiting the characteristics of these structures it is possible to develop an algorithm that is faster than the one shown in the previous section.

```
% INSERT INPUT HERE

% Initializations
trasf = [];
NW = [];
q_v = [];
```

```
Q_v = [];
NW_struct = [];
fcont = 0;
lcont = 1;

for u = [1:1:size(w,2)]
    NW = [NW; w{u}];
    fcont = size(w{u},1);
    q_v = [q_v; q{u}];
    Q_v = [Q_v; Q{u}];
    NW_struct{u} = [lcont:1:lcont+fcont-1];
    lcont = lcont + fcont;
end

[goodlist] = pyramidal(N, NW, NW_struct, Apriori, I, q_v, ...
                                                    Q_v, target_firm);
```

The function *pyramidal* allows computing the set of winning coalitions for the company *target_firm*. The function receives the same data of the function *compute_goodlist*.

```
function [goodlist] = pyramidal(N, NW, NW_struct, Apriori, I, q_v, Q_v, ...
                                                        target_firm)
count_good = 0;
count_list = [];
goodlist = {};

for j = [1:1:length(I)]
    coal = nchoosek(I([1:1:length(I)]),j);
    controlled = [];
    for h = [1:1:size(coal,1)]

        [Z_A] = A_Priori_Check(coal(h,:), N, Apriori);

        for i = [1:1:length(N)]
            if i>1
                tmp_comp = union(NW_struct{i},tmp_comp);
            else
                tmp_comp = NW_struct{1};
            end

            contr_tmp = (sum(sum(NW(NW_struct{i}, ...
                union(controlled(controlled>0), coal(h,:))),2) ...
                >=q_v(NW_struct{i}))>=Q_v(i))==1;

            if not(contr_tmp==0)
                controlled(i) = min(Z_A(i),find(contr_tmp)>0);
            else
                controlled(i) = 0;
            end
        end

        if not(isempty(controlled))
            if controlled(target_firm)>0
                count_good = count_good + 1;
                goodlist{count_good} = coal(h,:);
```

```
                count_list(count_good,:) = zeros(length(I),1);
                count_list(count_good,coal(h,:)-length(N)) = 1;
            end
        end
    end
end
```

The computational time required by this algorithm is lower than the one of the previous algorithm.

For example, consider the situation in which there are 10 companies and 15 investors and where every investor has some share in each company and where company k owns shares of all the companies from $k + 1$ to 10. Moreover, there is no voting agreement. In this case the algorithm took about two minutes.

In this chapter some algorithm for the application of the model has been presented. The aim of these algorithms is to provide a description of the situation.

CHAPTER 5

Further results in this thesis:

Some applications

In this chapter we present some applications of the model and of the algorithms introduced in the previous chapters. The first application concerns the development of a strategy that allow to increase the influence of an investor in a target company. The second application we present permits to identify the investors that who have a dominant influence on a target firm. Finally the third application is related to the fact that the model provides an accurate measurement of the power of each shareholder that allows to evaluate correctly the amount of private benefits deriving from the control of a company.

5.1 To increase the influence of an investor

In this section it is developed a method that allows to an investor to increase his influence in a target company by purchasing a certain amount of shares. However, the investor has not necessarily to purchase shares of the target company; in fact, he can buy shares of companies that are linked to the target company, directly or indirectly.

We consider the Banzhaf index suitable to measure the influence of an investor in a company; for further information see (Leech, 2002).

The advantages of this method are: the possibility of reducing the costs of the takeover by taking advantage of the different share prices of the other companies, the diminishing effect on the price of the shares of a massive purchase of securities and a more diversified investment.

It is important to note that in many jurisdictions, in the case of regulated markets, the direct or indirect acquisition of the shares of company over a certain threshold triggers the obligation to propose a tender offer. In this case it is necessary to modify the proposed algorithm.

In this section we propose a method applicable to markets in which there is no such obligation, such as the U.S. market. However, this method can be applied also to non-regulated markets since these markets do not usually apply the legislation on mandatory tender offer.

In addition of the input data presented in section 4.1, the proposed algorithms use also the following variables:

- *p_shares*: the vector of the share prices;
- *inv_player*: the investor that wants to increase his influence over the target company;
- *target_firm*: the target company.

For example, here is a possible instance of the data structure.

```
I = [3 4 5];
N = [1 2];

w{1} = [0 0 10 20 35; 0 0 10 40 20];
w{2} = [20 0 10 30 25];

q{1} = [50.01; 59.01];
q{2} = [50.01];

Q{1} = 2;
Q{2} = 1;

Apriori{1} =[1 1 2];        % In this case players 3 and 4 have signed
                            % a voting agreement
Apriori{2} =[1 2 3];


p_shares = [3 2 50];
inv_player = 4;
target_firm = length(N);
```

5.1.1 Pyramidal case

In this subsection it is developed an algorithm for the pyramidal control structures.

This algorithm allows to compute the Banzhaf index of the investor *inv_player* in the current situation and the maximum Banzhaf index that he can obtain through the purchase of the shares. Then the investor has to decide the target Banzhaf index that he wants to attain.

```
global p_shares;
global goodlist;
global count_good;
global NW;
global N;
global NW_struct;
global Apriori;
global I;
global q_v;
global Q_v;
global target_firm;
global inv_player;
global banzhaf_target;
global float;

% INSERT INPUT HERE

% Initializations
ris = 1;
trasf = [];
NW = [];
```

```
q_v = [];
Q_v = [];
NW_struct = [];
fcont = 0;
lcont = 1;

for u = [1:1:size(w,2)]
    NW = [NW; w{u}];
    Fcont = size(w{u},1);
    q_v = [q_v; q{u}];
    Q_v = [Q_v; Q{u}];
    NW_struct{u} = [lcont:1:lcont+fcont-1];
    lcont = lcont + fcont;
end

% Computes the float for each company
float = 100-sum(NW,2);

NW_tmp = NW;
NW_tmp(:,inv_player) = NW_tmp(:,inv_player);

% Computes the winning coalitions in the current situation
[good] = piramidale(N, NW_tmp, NW_struct, Apriori, I, q_v, Q_v, ...,
                                                    target_firm, inv_player);

% Initializiation
for i = [1:1:size(good,2)]
    los{i} = [];
end

% Computes the non-crucial players for every winning coalition
for i = [1:1:size(good,2)-1]
    for j = [i+1:1:size(good,2)]
        if all(ismember(good{i},good{j}))
            % good{i} is included in good{j}
            if isempty(los{j})
                los{j} = setdiff(good{j},good{i});
            else
                los{j} = union(los{j}, setdiff(good{j},good{i}));
            end
        end
    end
end

winnn = zeros(length(I),1);

% Computes the Banzhaf index
for i = [1:1:size(good,2)]
    los{i} = unique(los{i});
    win = setdiff(good{i},los{i});
    winnn(win-length(N)) = winnn(win-length(N)) + 1;
end
for j = [1:1:length(I)]
    banzhaf(j) = winnn(j)/sum(winnn);
end

% Computes the winning coalitions in the case in which the investor inv_player
% decides to buy all the float of all the companies
NW_tmp = NW;
NW_tmp(:,inv_player) = NW_tmp(:,inv_player) + float;
[good] = piramidale(N, NW_tmp, NW_struct, Apriori, I, q_v, Q_v, target_firm);
```

```
% Initialization
for i = [1:1:size(good,2)]
    los{i} = [];
end

% Computes the non-crucial players for every winning coalition
for i = [1:1:size(good,2)-1]
    for j = [i+1:1:size(good,2)]
        if all(ismember(good{i},good{j}))
            % good{i} is included in good{j}
            if isempty(los{j})
                los{j} = setdiff(good{j},good{i});
            else
                los{j} = union(los{j}, setdiff(good{j},good{i}));
            end
        end
    end
end

winnn = zeros(length(I),1);

% Computes the Banzhaf index
for i = [1:1:size(good,2)]
    los{i} = unique(los{i});
    win = setdiff(good{i},los{i});
    winnn(win-length(N)) = winnn(win-length(N)) + 1;
end
for j = [1:1:length(I)]
    banzhaf_b(j) = winnn(j)/sum(winnn);
end

disp('Banzhaf Indices');
disp(sprintf('%d :  %f           ', [1:1:length(I);banzhaf]));
disp(sprintf('Maximum Banzhaf Index for the selected investor: %f', ...
                                        banzhaf_b(inv_player - length(N))));

% Asks for the target level of the Banzhaf index
banzhaf_target = input('Banzhaf target: ');
if banzhaf_target>banzhaf_b(inv_player-length(N))
    disp('it is not possible to attain the target level');
    break
else
    if banzhaf_target<=banzhaf(inv_player-length(N))
    disp('the target level has been already attained');
    break
    end
end
```

Once the investor has decided the minimum level of the Banzhaf index to be attained, the investor has to run the optimization procedure. The goal of this procedure is to identify the amount of shares to be purchased that allows to attain the specified degree of influence and such that the costs are minimized.

Here is the objective function to be minimized.

```
function [f] = funobj(x)

global p_shares;
global goodlist;
global count_good;
global NW;
global N;
global NW_struct;
global Apriori;
global I;
global q_v;
global Q_v;
global target_firm;
global inv_player;
global banzhaf_target;
global float;

% MAX_CONTR is the penalty in the case in which the target Banhaf index is not
% attained. It must be greater that the cost of purchasing all the shares that
% belong to the float

MAX_CONTR = 1000000;

NW_tmp = NW;
NW_tmp(:,inv_player) = NW_tmp(:,inv_player) + x'.*(float/100);
[good] = piramidale(N, NW_tmp, NW_struct, Apriori, I, q_v, Q_v, target_firm);

% Initialization
for i = [1:1:size(good,2)]
    los{i} = [];
end

% Computes the non-crucial players for every winning coalition
for i = [1:1:size(good,2)-1]
    for j = [i+1:1:size(good,2)]
        if all(ismember(good{i},good{j}))
            % good{i} is included in good{j}
            if isempty(los{j})
                los{j} = setdiff(good{j},good{i});
            else
                los{j} = union(los{j}, setdiff(good{j},good{i}));
            end
        end
    end
end

winnn = zeros(length(I),1);

% Computes the Banzhaf index
for i = [1:1:size(good,2)]
    los{i} = unique(los{i});
    win = setdiff(good{i},los{i});
    winnn(win-length(N)) = winnn(win-length(N)) + 1;
end
for j = [1:1:length(I)]
    banzhaf(j) = winnn(j)/sum(winnn);
end

% Checks if the banzhaf index obtained with the purchasing is at least equal to
% the target
contr_x = not((banzhaf(inv_player-length(N))<banzhaf_target));
```

```
% Computes the value of the function
f = p_shares*(x'.*(float/100)) - contr_x*MAX_CONTR;
end
```

The particular form of the function imposes to use optimization methods that require only the evaluation of the function.

One of these methods is the Genetic Optimization. Briefly, it works by creating a set of possible solutions to the problem and updating that set on the basis of certain specifications. The optimization process continues until it meets a stop condition.

A solver of this kind is include in Matlab; using that solver it is possible to find an optimal solution x, i.e. the vector of the purchases to be made in the different companies. Notice that a necessary condition for a solution in order to be optimal is that the value $funobj(x)$ is not positive.

5.1.2 General case

Also in this case it is necessary to determine the Banzhaf index of investor *inv_player* in the current situation and the maximum Banzhaf index that he can attain. Then the investor has to decide the level of Banzhaf index that he wants to attain.

```
global p_shares;
global goodlist;
global count_good;
global NW;
global N;
global NW_struct;
global Apriori;
global I;
global q_v;
global Q_v;
global target_firm;
global inv_player;
global banzhaf_target;
global float;
global banzhaf_target;

% INSERT INPUT HERE

% Initializations
trasf = [];
NW = [];
q_v = [];
Q_v = [];
NW_struct = [];
fcont = 0;
lcont = 1;
```

```
for u = [1:1:size(w,2)]
    NW = [NW; w{u}];
    fcont = size(w{u},1);
    q_v = [q_v; q{u}];
    Q_v = [Q_v; Q{u}];
    NW_struct{u} = [lcont:1:lcont+fcont-1];
    lcont = lcont + fcont;
end

% Computes the float for each company
float = 100 - sum(NW,2);

NW_tmp = NW;
NW_tmp(:,inv_player) = NW_tmp(:,inv_player);

% Computes the winning coalitions in the current situation
[good] = calcola_goodlist (N, I, NW_tmp, Apriori, NW_struct, q_v, Q_v, ...
                                                              target_firm);

% Initialization
for i = [1:1:size(good,2)]
    los{i} = [];
end

% Computes the non-crucial players for every winning coalition
for i = [1:1:size(good,2)-1]
    for j = [i+1:1:size(good,2)]
        if all(ismember(good{i},good{j}))
            % good{i} is included in good{j}
            if isempty(los{j})
                los{j} = setdiff(good{j},good{i});
            else
                los{j} = union(los{j}, setdiff(good{j},good{i}));
            end
        end
    end
end

winnn = zeros(length(I),1);

% Computes the Banzhaf index
for i = [1:1:size(good,2)]
    los{i} = unique(los{i});
    win = setdiff(good{i},los{i});
    winnn(win-length(N)) = winnn(win-length(N)) + 1;
end
for j = [1:1:length(I)]
    banzhaf(j) = winnn(j)/sum(winnn);
end

% Computes the winning coalitions in the case in which the investor inv_player
% decides to buy all the float of all the companies
NW_tmp = NW;
NW_tmp(:,inv_player) = NW_tmp(:,inv_player) + float;
[good] = calcola_goodlist (N, I, NW_tmp, Apriori, NW_struct, q_v, Q_v, ...
                                                              target_firm);

% Initialization
for i = [1:1:size(good,2)]
    los{i} = [];
end
```

```
% Computes the non-crucial players for every winning coalition
for i = [1:1:size(good,2)-1]
    for j = [i+1:1:size(good,2)]
        if all(ismember(good{i},good{j}))
            % good{i} is included in good{j}
            if isempty(los{j})
                los{j} = setdiff(good{j},good{i});
            else
                los{j} = union(los{j}, setdiff(good{j},good{i}));
            end
        end
    end
end

winnn = zeros(length(I),1);

% Computes the Banzhaf index
for i = [1:1:size(good,2)]
    los{i} = unique(los{i});
    win = setdiff(good{i},los{i});
    winnn(win-length(N)) = winnn(win-length(N)) + 1;
end
for j = [1:1:length(I)]
    banzhaf_b(j) = winnn(j)/sum(winnn);
end

disp(sprintf('Banzhaf Indices\n'));
disp(sprintf('%2.0f :  %f \n', [1:1:length(I);banzhaf]));
disp(sprintf('Maximum Banzhaf Index for the selected investor: %f',
banzhaf_b(inv_player - length(N))));

% Asks for the target level of the Banzhaf index
banzhaf_target = input('banzhaf target: ');
if banzhaf_target>banzhaf_b(inv_player-length(N))
    disp('it is not possible to attain the target level');
    break
else
    if banzhaf_target<=banzhaf(inv_player-length(N))
    disp('it has been already attained the target level');
    break
    end
end
```

The next step is the identification of the amount of shares to be purchased in order to attain the target level of the Banzhaf index and such that the costs are minimized.

Here is the objective function to be minimized.

```
function [f] = funobj(x)
global p_shares;
global goodlist;
global count_good;
global NW;
global N;
global NW_struct;
global Apriori;
```

```
global I;
global q_v;
global Q_v;
global target_firm;
global inv_player;
global banzhaf_target;
global float;

% MAX_CONTR is the penalty in the case in which the target Banhaf index is not
% attained. It must be greater that the cost of purchasing all the shares that
% belong to the float
MAX_CONTR = 1000000;

NW_tmp = NW;
NW_tmp(:,inv_player) = NW_tmp(:,inv_player) + x'.*(float/100);

[good] = calcola_goodlist (N, I, NW_tmp, Apriori, NW_struct, q_v, Q_v, ...
                                                              target_firm);

% Initialization
for i = [1:1:size(good,2)]
    los{i} = [];
end

% Computes the non-crucial players for every winning coalition
for i = [1:1:size(good,2)-1]
    for j = [i+1:1:size(good,2)]
        if all(ismember(good{i},good{j}))
            % good{i} is included in good{j}
            if isempty(los{j})
                los{j} = setdiff(good{j},good{i});
            else
                los{j} = union(los{j}, setdiff(good{j},good{i}));
            end
        end
    end
end

winnn = zeros(length(I),1);

% Computes the Banzhaf index
for i = [1:1:size(good,2)]
    los{i} = unique(los{i});
    win = setdiff(good{i},los{i});
    winnn(win-length(N)) = winnn(win-length(N)) + 1;
end
for j = [1:1:length(I)]
    banzhaf(j) = winnn(j)/sum(winnn);
end

% Checks if the Banzhaf index obtained with the purchasing is at least equal to
% the target
contr_x = not((banzhaf(inv_player-length(N))<banzhaf_target));

% Computes the value of the function
f = p_shares*(x'.*(float/100)) - contr_x*MAX_CONTR;

end
```

Also in this case it is necessary to use optimization methods that require only the evaluation of the function. It is possible to use the same solver presented in the previous subsection.

Thus it is possible to find an optimal solution x, i.e. the vector of the purchases to be made in the different companies. Notice that also in this case a necessary condition for a solution in order to be optimal is that the value $funobj(x)$ is not positive.

5.2 To identify those who have a dominant influence

One of the applications introduced in the third chapter was the identification in an objective way of the investors who have a dominant influence over a given company.

To identify those investors, it is useful to compute the Banzhaf power indices of all players in every company. Then those that have a high Banzhaf index can be considered to be influential.

Here we propose a procedure that allows to compute those indices.

```
% INSERT INPUT HERE

% Initializations
NW = [];
q_v = [];
Q_v = [];
NW_struct = [];
fcont = 0;
lcont = 1;

for u = [1:1:size(w,2)]
    NW = [NW; w{u}];
    fcont = size(w{u},1);
    q_v = [q_v; q{u}];
    Q_v = [Q_v; Q{u}];

    % NW_struct indicates which rows of NW are related to company u
    NW_struct{u} = [lcont:1:lcont+fcont-1];
    lcont = lcont + fcont;
end

% Computes the winning coalitions for all companies
for target_firm = [1:1:length(N)]
    winn{target_firm} = calcola_goodlist(N, I, NW, Apriori, NW_struct,  ...
                                                    q_v, Q_v, target_firm);
    good = winn{target_firm};

    % Initialization
    for i = [1:1:size(good,2)]
        los{i} = [];
    end

    % Computes the non-crucial players for every winning coalition
```

```
    for i = [1:1:size(good,2)-1]
        for j = [i+1:1:size(good,2)]
            if all(ismember(good{i},good{j}))
                % good{i} is included in good{j}
                if isempty(los{j})
                    los{j} = setdiff(good{j},good{i});
                else
                    los{j} = union(los{j}, setdiff(good{j},good{i}));
                end
            end
        end
    end

    winnn = zeros(length(I),1);

    % Computes the Banzhaf index
    for i = [1:1:size(good,2)]
        los{i} = unique(los{i});
        win = setdiff(good{i},los{i});
        winnn(win-length(N)) = winnn(win-length(N)) + 1;
    end
    for j = [1:1:length(I)]
        banzhaf(j) = winnn(j)/sum(winnn);
    end

    disp(sprintf('Company %d\n',target_firm));
    disp(sprintf('Banzhaf Indices\n'));
    disp(sprintf('%2.0f :  %f \n', [1:1:length(I);banzhaf]));
end
```

This algorithm also allows conducting some scenario analysis; in fact, it is possible to observe how the set of the winning coalitions changes varying the weights of the players and the voting agreements among investors.

5.3 Application to Corporate Finance

In this section we propose an application of the model to Corporate Finance.

The existence of private benefits that can be extracted by those who control a company is proved by several studies; see f.i. (Nenova, 2003). It is also assumed that such private benefits are linked to the value of the voting rights of the control-block.

A possible approach to compute the value of voting rights is presented in (Nenova, 2003); in this paper the normalized Shapley value is considered as a measure of the decision power of each shareholder. However if one computes that value using only the information about the directed shareholders, it is not possible to take into account the effects of voting agreement and of cross-ownerships.

The method introduced in this thesis allows to identify who effectively control every company and thus it allows to compute correctly the value. However, in order to use this method it is necessary to change the assumptions made about the float; as a consequence, some minor modifications of the algorithm are needed.

Once the influence of each investor is properly measured, it is possible to compute more accurately the value of the voting rights of the control-block.

In this chapter we have shown some algorithms that allows to solve the proposed problems.

CHAPTER 6
Conclusions regarding part 1

The object of the analysis of this first part of the thesis was the representation of corporate control.

First we presented a model that allows to take into account voting agreements and different types of voting systems. Then it has been developed a method for the identification of the investors who control each company.

Then three algorithms have been presented in order to apply the model to real cases. The first algorithm can be applied in every case and provides solutions even when the other methods cannot. The second one is also applicable to the situations in which there are cross-ownerships, but it can not identify all the reductions; however it is faster than the first algorithm. The computational time can be further decreased by using the third algorithm, which is valid only for pyramidal structures.

Finally, some applications have been presented together with some algorithms.

Despite the results presented in this thesis, there are two important research directions that can be developed.

First it is useful to develop faster algorithms, especially for the case of cross-ownerships. One way is to develop a method able to identify the different structures, pyramidal or cyclic, in a given market. Then one can apply the algorithm that is more suitable to the various structures in order to exploit, where possible, the speed of the algorithm for pyramidal structures.

Finally it is possible to compute the value of the voting rights in a given market using the method here proposed following the approach of (Nenova, 2003).

PART 2
SCORING METHOD THAT ARE ROBUST TO COLLUSION

CHAPTER 1
Pre-existing situation

In this chapter we present a problem that is examined in the second part of thesis and a specific method for solving it.

1.1 Area of application of the Collusion-Robust evaluation methods

Some goods may not have a market from which one can obtain their values and thus in order to evaluate such goods it may be necessary to resort to the valuations of some experts.

For example, consider the case in which several individuals own some assets that if used in combination allow to realize a certain project whose revenues will be divided in proportion to the contributions. The value of the assets owned by each individual can be determined through their evaluation by all the contributors.

Obviously every individual may have interest in overstating the value of his assets and in underestimating the value of the other assets. Thus it is necessary to find a method that allows to reduce the risk of manipulation of the evaluations.

1.2 Specific application of the Collusion-Robust evaluation methods: Euribor rates

The Euribor rates are determined by the following procedure. To every bank that belong to a certain panel it is required to provide "the daily quotes of the rate, rounded to three decimal places, that each panel bank believes one prime bank is quoting to another prime bank for interbank term deposits within the euro zone".

Then for each maturity the highest and lowest 15% of the quotes are discarded and the remaining quotes are averaged to obtain the Euribor rate for such maturity.

A similar procedure is also used for the determination of the Eoniaswap and the Eurepo rate. For those who are interested in the subject, more information is available at http://www.euribor-ebf.eu/assets/files/Euribor_tech_features.pdf.

Notice that in this application it has been used the trimmed mean, but instead one can use other methods such the one presented in the next section.

1.3 Collusion-Robust evaluation method: the Coherent Majority Average

In this section we present a method to address the problem introduced in the previous section: the Coherent Majority Average. This method has been published in (Gambarelli, 2008).

The method assumes that the majority of the scores are reliable and it discards all the scores that are not close enough to the majority of the evaluations.

Let be E the set of elements to be evaluated and $J = [1, \ldots, n]$ the set of players. For every $e \in E$, let be $v(e)$ the set of evaluations given to it. Let be $o(e)$ the ordered set of the scores in $v(e)$.

The procedure to compute the CMA for an element $e \in E$ is the following:

- compute the majority of players m as the integer part of $(n/2)+1$;
- for each cluster of m scores that belong to $o(e)$, calculate the difference between the highest and the lowest score;
- if the minimum of these differences is positive, then the CMA is the arithmetic mean of the scores that belong to the clusters with minimal difference;
- if the minimum of these difference is zero, then the majority of players give the same score z. In this case the CMA is the arithmetic mean of the scores that are between $z - i$ and $z + i$, where i is the minimum tick of the scoring system.

Finally, it is important to notice that in some cases the Coherent Majority Average is more robust to manipulation than other methods, as illustrated in the following example.

Assume that there are seven players that have to give to each element some valuations. Assume also that two players collude in order to influence the results by assigning the maximum score (f.i. 10) and that the others assign the scores: 2, 3, 4, 5, 5.

In this case if the value of the elements is computed using a trimmed mean that excludes the maximal and the minimal evaluation, the two players have been able to influence the result. Using the CMA the two players cannot manipulate the result giving the maximum score. However the two players, by reducing their valuations, may still be able to influence the result. Notice thus that the CMA is able to eliminate or to reduce the effect of some types of collusion, while the trimmed mean seems less robust.

These considerations suggest that for some applications, such as the determination of the Euribor rates, it is better to use the Coherent Majority Average.

CHAPTER 2

My published results: the Anti-collusion Average

In this chapter we present the Anti-collusion average, a new Collusion-Robust scoring method, introduced in (Bertini, Gambarelli and Uristani, 2010).

The objective of this method is the identification of the players that are more colluded in order to discard their evaluations. To do this it is necessary to take into account the whole set of evaluations. No assumptions on the behavior or on the preferences of the players are made, apart the fact that a player prefers to obtain a high evaluation of his elements rather than a low one.

The method allows taking into consideration that the players can form coalitions to influence their evaluations.

The computation of the Anti-collusion average is based on the computation of two types of collusion indices, one for each group of players and one for each individual player. We assign a coalitional index to each coalition on the basis of the evaluations given by each player. Then we assign an individual collusion index to each player that is equal to the maximum coalitional index of the coalitions to which the player belongs. Finally we select the most reliable players (i.e. those with the lowest individual collusion index). The ACA of an element is the average of the evaluations given by the players to such element.

In the following sections we present the method of the ACA in detail.

2.1 Indices of Collusion

The first step is to compute the collusion indices that allow detecting the most colluded players.

The set of players is divided in:

- $J = [1,\ldots, n]$, the players who own at least an element;
- $\underline{J}$, the other players.

The elements that have to be evaluated are divided into two sets:

- the set E contains the elements owned to the players that belong to J;
- the set $\underline{E}$ contains the elements that are not owned by any player.

Let us define P the set of the coalitions of players (excluding the global one). For each $p \in P$, we define p' the complement of p with respect to P.

Let be Π a finite set of positive rational numbers.

We define the evaluations' function a the function defined on $(J \cup \underline{J}) \times (E \cup \underline{E})$ with values in $\Pi \cup \varnothing$; such function assigns to each element a set of evaluations.

To obtain the coalitional collusion indices, we define:

- $k(x, y)$ is the cardinality of the set of the numeric scores that players that belongs to x assign to all elements belonging to all players corresponding to y;
- $m(x, y)$ is the function that assigns to all the elements belonging to y the arithmetic mean of the scores given to them by the players in coalition x if $k(x, y)>0$ and 0 otherwise.

We denote by $\underline{x}$ the union between $\underline{E}$ and all the elements that belong to x.

For each $p \in P$ we define the index of valuation of the coalition x:

$$\mathrm{I}(x) = \begin{cases} 1 & \text{if } m(p, \mathrm{x}) = 0 \text{ or } m(p', x) = 0 \\ m(p, x)/m(p', \mathrm{x}) & \text{elsewhere} \end{cases}$$

The coalitional collusion index $r(p)$ of each coalition $p \in P$ is equal to the ratio between $I(p)$ and $I(\underline{p}')$.

We restrict the computation of the coalitional index to the coalitions with cardinality smaller than $n/2$ (denoted by P') since we assume that bigger coalitions are not colluded.

For each player j we define the individual collusion index $c(j)$ as the maximum value of $r(p)$ for which all $p \in P'$ to which j belongs.

2.2 Anti-Collusion Average

In the previous section we have shown how to compute the collusion indices. On the basis of such indices, in this section we give the definition of the Anti-Collusion Average.

On the basis of the individual collusion indices we divide the players into reliability classes. We denote with C_1 the set of players j such that $c(j)$ is minimum. For each integer $h > 1$ we call C_k the set of players that belong to $(J \cup \underline{J}) \setminus (C_1 \cup ... \cup C_{k-1})$ with minimum collusion index.

Finally we build the set R_e of the players whose evaluations of element e have to be considered. Such set contains the set of players that gave a valuation to such element and that belong to the first reliability class. If the cardinality of such set is less than $n/2$, we add to this set the players that belong to the second reliability class. If the cardinality of the set obtained in such way is less than $n/2$ we proceed analogously with the third reliability class and so on until the cardinality of the set R_e is greater or equal to $n/2$.

The Anti-collusion average of element e is the arithmetic mean of the evaluations given to such element by the players belonging to R_e.

CHAPTER 3

Further results in this thesis

In this chapter we show a procedure that allows to evaluate the effectiveness of the different Collusion-Robust methods in reducing the collusion among players.

As it has been shown in the previous chapter, there exist different methods for reducing the influence of the possible collusions on the evaluations.

Since there are many methods that allow dealing with the same problem, it is natural to ask which method is the most suitable and, in general, what are their performances; to this end we propose a method based on Game Theory. This method can be applied to any type of evaluation method with the only condition that each player must own at least an element that has to be evaluated.

3.1 Analysis method

The method that we present for the comparison of the different Collusion-Robust methods is based on Cooperative Games. In this way it is possible to decide which method is the most effective for analyzing a particular situation.

The reason for which we use Game Theory is that we consider that the players can form coalitions in order to manipulate the results in a way that favours them; thus it is assumed that such players can coordinate their evaluations. Hence we can represent this situation with a game in characteristic function form.

More precisely, we define a game in characteristic function form (N, v) where:

- N is the set of players;
- $v(S) = V(S) \qquad \forall S \subseteq N$.

$V(S)$ is the maximum sum of valuations given to the elements that belong to the players of coalition S when they coordinate their valuations. Moreover, it is assumed that the colluding players know the valuation of the others; in such way we can verify the robustness of the method in the worst case.

After calculating the values of the characteristic function, we verify if it is convenient for the players to form coalitions; in these cases there are incentives for colluding.

A first criterion is to verify if the game is subadditive; in such case it is not convenient for the players to cooperate and thus the analyzed method does not offer, in this case, any incentive to collude.

A second criterion is to verify if the game is inessential. Even in that case there are no incentives to collude.

If the game is neither subadditive nor inessential, then it is possible that there are some incentives to collude. However in this case it is not sure if it exist a stable allocation. As a consequence, a third criterion to determine if a method is better than the others is to verify if the game computed on the basis of the such method has an empty core, while the one computed for the others is not empty.

Finally, if on the basis of the previous criteria it is not possible to decide which method is the best one, then for each method we compute the value $\max_S v(S)$. The method for which such value is the minimum is to be preferred to the others.

The analysis of a particular situation can be done through simulations. After identifying the number of players and the number of elements, we generate a set of evaluations on the basis of some probabilistic assumptions and then we compute the values of the characteristic function of the game. Then we apply the criteria presented above. After making a certain number of simulations, we identify the method that was better than the others more frequently.

3.2 Some results

In this section we analyze the following example. Consider the case in which there are five players and five elements that have to be evaluated, one for each player. The evaluations given to the elements belong to the interval [1, 10].

We examined the performance of three methods: the ACA, the CMA and the trimmed mean that discards 30% of the evaluations.

On the basis of the method presented in the previous section, we made 500 simulations of possible evaluations.

The Coherent Majority Average has proved to be the best in the 41% of the cases and the second-best method in the 44% of the cases; only in the remaining 15% of cases it was worse than the other two methods.

The trimmed mean has been the best method in the 32% of the cases and the second best in the 25% of the cases; however in the 43% of cases it was the worse method.

Finally, the Anti-collusion average is the best method in the 26% of the cases and the second best in the 32% of cases.

Thus in this situation the Coherent Majority Average is better than the other two.

In this chapter we presented a method for the comparison of different Collusion-Robust methods and it has been identified the most suitable method in a specific situation.

CHAPTER 4
Conclusions regarding part 2

In this second part of the thesis we presented a Collusion-Robust evaluation method and we suggested an application of such methods for determining the Euribor, Eoniaswap and Eurepo rates.

The existence of different evaluation methods made necessary to find a procedure that allows identifying the method that is the most suitable in a particular situation. To this end we developed a method for the analysis of the performances of different Collusion-Robust methods.

A limitation of this method is the high amount of time required to obtain the results; a reduction of such time is a possible further development. In such way it could be possible to verify in a reasonable time which Collusion-Robust method is the most suitable for the determination of the Euribor rates.

GENERAL CONCLUSIONS OF THE THESIS

In many situations the ability of a player to influence a result does not depend only on his strength, but also on the relationships that he establish with the other players. To take into consideration the effects of cooperation (or collusion) in two applied contexts, we have developed some methods based on Cooperative Game Theory.

In the first part of the thesis we considered a market in which there are several companies of different kind. Moreover each company may be owned by investors and by other companies and investors are allowed to cooperate using voting agreements.

Given this situation, the main question was: which coalitions of investors can control a certain company? To provide an answer we developed a new model and we presented some cases: the case of golden shares and of cooperative companies.

We considered two types of representation of relationships among players: the coalitional structure and the graph-restricted communication structure. We argued that the former is to be preferred in this application.

After having defined the model, our aim was to develop some algorithms to carry out the computations by using only publicly available data.

The first algorithm allows to compute the reduced extensions, while the second one allows to compute the set of coalitions of investors that have the control when there are cycles. Finally we presented a third algorithm, faster than the previous ones, designed for pyramidal structures. This last algorithm was necessary since the computational time of the other methods can be very high for medium instances.

Then we presented some other applications of the model.

The first application concerns the identification of the investors who have a dominant influence on the various companies. In this case we have also provided an algorithm for the computations.

The second application concerns the identification of strategies that allows to a specific investor to increase his influence in a company. Also in this case we have provided an algorithm for the computations.

Then we presented a possible application to Corporate Finance.

Finally we have presented some further research directions.

Cooperation can be convenient for the players in many cases. However, in some cases the cooperation among players is not to be considered a good thing as, for example, in the case in which a group of players have to provide some evaluations. In such cases it may be important that they do not collude.

In the second part of the thesis we analyzed the problem of evaluating a certain object on the basis of the valuations of different subjects that may collude. This kind of problems can be found in the financial field in the case of the determination of the Euribor rates.

Also in this case the relationships among the players have an important role and it is necessary to take them in account since they can influence the valuations of the different objects.

The problem can be dealt with different Collusion-Robust methods. Since many methods have been developed, it was necessary to identify which method is the most suitable for a given situation. To do so we presented an analysis method, based on the Cooperative Game Theory, that identifies which Collusion-Robust method minimizes the incentives to collude.

REFERENCES

Annunziata, F. (2004). *La disciplina del mercato mobiliare*, Giappichelli editore, Torino

Aumann, R. and Drèze, J. (1974). "Cooperative games with coalition structure", *International Journal of Game Theory*, 3, 217-237

Banzhaf, J. (1965). "Weighted Voting Doesn't Work: A Mathematical Analysis", *Rutgers Law Review*, 19, 317-343.

Bertini, C., G. Gambarelli and A. Uristani (2010). "Collusion Indices and an Anti-collusion Average" S. Greco, R. Marques Pereira, *Preferences and decisions: models and applications*, in Studies in Fuzziness and Soft Computing, Springer Verlag.

Crama, Y. and Leruth, L. (2007). "Control and voting power in corporate networks: Concepts and computational aspects", European Journal of Operational Research, 178, 879-893.

Coleman, J. S. (1971). "Control of Collectivities and the Power of a Collectivity to Act", *Social Choice*, Lieberman, New York.

Denti, E. and Prati, N. (2001). "An algorithm for winning coalitions in indirect control of corporations", *Decisions in Economics and Finance*, 24, 153-158.

Gambarelli, G. (2008). "The 'Coherent majority average' for juries' evaluation processes", Journal of Sport Sciences, 1-5

Gambarelli, G. (2003). *Giochi Competitivi e Cooperativi*, 2ª ed. Torino: Giappichelli Editore

Gambarelli, G. and Owen, G. (1994). "Indirect control of corporations", *International Journal of Game Theory*, 23, 287-302

Gambarelli, G. and Uristani, A. (2009). "Multicameral Cohesion Voting Games", *Central European Journal of Operations Research,* Volume 17, Issue 4, 433-460

Gambarelli, G. and Palestini, A. (1999). "Minimax Multi-District Apportionments", *Homo Oeconomicus*, 24, 335-356.

Gambino, A. and Santosuosso, D. U. (2006). *Società di capitali*, Giappichelli editor, Torino

Greene, E. F. (2006). U.S. regulation of the international securities and derivative markets, Volume 1, Aspen Publishers

Khmelnitskaya, A.B. (2007). "Values for graph-restricted games with coalition structure", Memorandum 1848, Department of Applied Mathematics, University of Twente, Enschede, The Netherlands.

Leech, D. (2002). "An Empirical Comparison of the Performance of Classical Power Indices", *Political Studies*, 50, 1, pp.1-22.

Levy, M. (2007). "Control in Pyramidal Structures", No 07-08.RS, DULBEA Working Papers, ULB -- Universite Libre de Bruxelles.

Myerson, R. (1977). "Graph and cooperation in games", *Mathematics of Operation Research*, 2, 225:229.

Nenova, T. (2003). "The value of corporate voting rights and control: A cross-country analysis", *Journal of Financial Economics*, 68, 325-351.

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behaviour*, 3rd ed. 1953. Princeton, NJ: Princeton Academic Press.

Owen, G. (1977). "Values of games with a priori unions", Mathematical Economics and Game Theory, Springer, 76-88.

Owen, G. (1995). *Game Theory*, 3rd ed. San Diego:Academic Press.

Penrose, L. S. (1946). "The elementary statistics of majority voting", *Journal of the Royal Statistical Society*, 109, 53-57.

Shapley, L. S. (1953). "A Value for *n*-Person Games" in H.W. Kuhn and A.W. Tucker, *Contributions to the Theory of Games*, 2, 39, Princeton University Press, 307-317.

Shapley, L. S. and Shubik, M. (1954). "A Method for Evaluating the Distributions of Power in a Committee System", *American Political Science Review*, 48, 787-792.

Slikker, M. and Van Den Nouweland, A. (2000). *Social and economic networks in cooperative game theory*, Kluwer Academic Publishers.

Uristani, A. (2007). *Applicazioni Politiche e Finanziarie dei Giochi Multicamerali*, 2nd Level Degree Thesis.