On aggregate and comparison functions for Motus/Lingo playing

Nathalie Chetcuti-Sperandio, Fabien Delorme, Sylvain Lagrue

On aggregate and comparison functions for Motus/Lingo playing

Nathalie Chetcuti-Sperandio, Fabien Delorme, and Sylvain Lagrue

CRIL Univ. Artois & CNRS, F-62300 Lens, France,
chetcuti|delorme|lagrue@cril.fr,
Home pages: http://www.cril.univ-artois.fr/~chetcuti|delorme|lagrue

Abstract Motus (Lingo) is a TV show based on a word game where each player has to guess a word, of which she knows only the length and the first letter. To do this, the player makes different proposals and she is given some hints based on the positions of the letters. We first show in this article that there is no strategy to find all the words with the available number of proposals. Next, we study different strategies based on aggregate functions and tie-breaks. All these strategies were tested on all the 5 to 10 letter French words, which finally leads us to discuss the respective performance of these strategies. We show that the criteria of average number of tries used, worst case and number of fails are conflicting.

1 Introduction

Motus is a TV word guessing game show aired since 1990 in France. It is the adaptation of the American TV game show Lingo, which was first broadcast in 1988. It combines phases of chance (close to bingo) and word searching phases, in the spirit of Mastermind. If the latter was studied [3,5], there has been no work on Motus to our knowledge. The major difference between both games is that the words considered in Motus come from dictionaries and do not form a homogeneous search space contrary to codes in Mastermind.

In this article, we focus on the last part of the game, where a player must guess 10 words of which she knows only the length and the first letter, in a given time. To do this, the player proposes various words and is notified of correctly-placed and incorrectly-placed letters. For each word, the player is entitled to six proposals at most. If she fails, a new word is proposed to her, until the time expires or until she guesses ten words. So the goal of the player is to fail as seldom as possible (otherwise all the time spent to look for the right answer will have been spent for nothing), while using as few proposals as possible (the allotted time is too short for the player to guess all the words using the granted six proposals each time).

This article presents various original experimental results. We first show that there is no strategy to find all the words with the granted number of proposals. Then, we study different strategies based on aggregate and tie-break functions.

These strategies were tested on all 5 to 10 letter words of the official French Scrabble dictionary (version 5), which finally enables us to discuss the respective performance of these strategies using a sizable number of words. In particular we show that the criteria of average number of tries used, worst case and number of fails are conflicting.

This article connects game resolution with techniques that are well known to the community. The problem considered here is a problem of decision making with incomplete information. The proposed solutions are based on well-known aggregate and comparison methods which are used, for example, for merging uncertain information.

2 Rules of the game and working assumptions

2.1 Rules of the game

We introduce here the rules of the game on which we rely in the remainder of this article. We focus on a two-player game with incomplete information. The first player (the "environment") does only one action: choosing one word among a set of authorized words. As regards the French TV game show, all the words from French reference dictionaries (Larousse and Robert) are authorized, except proper nouns, compound words and conjugated verbs (only present infinitive and past and present participles are accepted).

Once a word is selected, the number of letters as well as the first letter are given to the second player (the "contestant"), simply called "the player" in what follows. This one has a limited number of tries to guess the word (5 for 5-letter words, 6 for 6 to 10-letter words). For a proposal to be valid, it must be part of the reference dictionary, start with the first letter of the word to guess and have the correct number of letters in length.

After each proposal, the player is shown the correctly-placed letters (i.e. letters that are part of the word to guess and located in the same place in this word) and incorrectly-placed letters (i.e. letters that are part of the word to guess but located elsewhere in the word). If the player guesses the right word within the given number of tries, she wins, otherwise she loses.

Example 1 *The word to guess is the word "logique". The player is informed that she must guess a seven-letter word beginning with an L. She proposes the word "légales" first:*

Both words contain an L in the first position and a G in the third position: these letters are correctly placed. Moreover, both words contain the letter E, but at different locations (in the second and the sixth positions in the word "légales", in the seventh position in the word "logique"): the letter E is incorrectly placed. As the word to guess contains only one occurrence of this letter and the word

provided by the player contains two, it is considered, by convention, that only the first occurrence of the letter E is incorrectly placed. The other letters (A, L and S as well as the second occurrence of the letter E) are considered to be missing from the word to guess.

Then the player proposes the word "livides":

The letters in the first and the fourth positions are correctly placed, the letter E in the sixth position is incorrectly placed and the the other letters are missing. It should be noted that the letter I in the second position, although present in the word to guess at another place, is not part of the incorrectly-placed letters since another occurrence of the letter has already been indicated as being correctly placed.

The player is now able to guess the word: "logique":

From these basic rules, many variations can be applied. In particular the player can be encouraged to guess the word in a minimum of tries, whether by granting her more points if she answers in a minimum of tries, or by giving her a list of words to guess in a limited time.

2.2 Working assumptions

Given the availability of the different dictionaries, the set of *valid* words is built from the "official Scrabble dictionary", which has a much greater number of words than permitted by the original rules. Indeed it includes conjugated verbs, as well as some rare or obsolete words that are not part of the usual dictionaries.

We also consider that the words of this reference dictionary have the same probability of occurrence. This last assumption is not true in the TV game show where only words likely to be known by the players are selected. An algorithm could take advantage of this specificity, assuming that the words most present in a corpus of reference texts are more likely to be selected by the environment, but such an assumption is beyond the scope of this article.

3 Game modelling

We present in this part a modelling of the problem and an alternative game, "Evil Motus", which leads to an interesting result from the point of view of game theory. In this context, an evil genius can change the word after each proposal of the player.

First of all, we start with a more formal modelling of the game.

3.1 Definitions

Let D be the set of valid words. A word $w \in D$ is seen as a vector of letters where w_i represents the ith component of w.

D_l^n represents the set of valid n-letter words beginning with the letter l.

$BP(w, w') = \{i : w_i = w'_i\}$ represents the set of the indices of the correctly-placed letters.

$MP(w, w') = \{i \notin BP(w, w') : \exists j \neq i : w_i = w'_j,\ j \notin BP(w, w')$ *and* j *has not been used yet*$\}$ represents the set of the indices of the incorrectly-placed letters in w compared to w'. Since several solutions are possible for $MP(w, w')$, we consider only the minimal element from a lexicographical point of view. That is, if the same letter is used only once in the word to guess but appears several times in the proposal, we will consider only the lowest index (index of the first occurrence of the letter in the proposal).

Example 2 *Consider the following dictionary:*

$D_L^7 = \{LEGALES, LIGNIEZ, LIGNINE, LIGNITE, LIGUIEZ, LIVIDES, LOGIONS, LOGIQUE\}$.

Then $BP(LEGALES, LOGIQUE) = \{1, 3\}$ *and* $MP(LEGALES, LOGIQUE) = \{2\}$.

One can notice that BP is symmetric while MP is not.
Indeed, $MP(LOGIQUE, LEGALES) = \{7\}$

In order to compute the player's reasoning, we model her knowledge as follows: the player gave a set of proposals S and for each w element of S, she knows $BP^S(w)$ the set of the indices of the correctly-placed letters in w and $MP^S(w)$ the set of the indices of the incorrectly-placed letters in w. Then the subset of D_l^n of *possible* words (words likely to be the word to guess) is defined as $Poss(D_l^n, S, BP^S, MP^S) = \{w \in D_l^n : \forall w' \in S, BP^S(w') = BP(w, w')$ and $MP^S(w') = MP(w, w')\}$, i.e. the set of words matching all the words proposed by the player in terms of correctly and incorrectly-placed letters.

3.2 Evil Motus

From a game theory point of view, "Evil Motus" is a two-player asymmetric game with incomplete information. The rules are as follows: after "Evil Motus" chooses a first word, the player makes a proposal. "Evil Motus" has the right to change the word to guess, as long as the newly selected word has the same correctly-placed and incorrectly-placed letters for all the previous proposals: the information previously given is never changed and remains valid. Then the player is shown the correctly-placed and the incorrectly-placed letters for her last proposal.

Example 3 *Evil Motus chooses the word "logique". Thus it indicates to the player that she has to find a 7-letter word beginning with the letter L. The player proposes the word "légales". Evil Motus then indicates that the first and third letters are correctly placed, while the first E is incorrectly placed:*

Then the player proposes the word "livides". If it keeps the same word, Evil Motus will have to indicate that the L and the second I are correctly placed, while the E is incorrectly placed:

The only possible word would then be the word "logique" and, if the player is insightful, she will find it at the next try. Evil Motus therefore chooses to change the word to guess, and chooses the word "lignine". Then it indicates that, in the proposal made by the player (the word "livides") the first two letters are correctly placed, while the second I and the E are incorrectly placed:

There remain two possible words (namely "lignine" and "lignite"). Then the player proposes the word "lignine". If it keeps the same word, Evil Motus lost. So it chooses the word "lignite" and indicates that all the letters are correctly placed, except the penultimate one (the letter N) which is not present in the word:

The player will certainly guess the word at the next try, but no other word would have delayed her victory. It took the player four tries to guess the word.

Using a Minimax-type algorithm [8] for each initial letter and each length of words, one can determine the minimal number of words a player will have to propose to be sure to guess the word. We call this value the EMI (*Evil Motus Index*). If an EMI is greater than the number of tries available to the player to guess the word, it is possible to assert that no strategy ensures to guess all the words systematically. On the contrary, if the EMI is lower than the number of tries granted to the player, it means that there is at least one strategy for guessing all the words, given an initial letter and a length.

Table 1 gathers all the EMI found for all 5 to 10 letter words. For each cell of the table, it took between 1s and 3 days of calculations (program in Go language on Intel i7 3.1GHz, 16GB of RAM), the average duration being about 1 hour.

Some values deserve to be highlighted. For example the column of the letter X gives an EMI of 2 for a length of 7 letters or more. This ensures the player to guess the word at the next proposal after choosing a first "good" word. Some of these good words are the following ones (there may be others):

7 letters: XIMENIA
8 letters: XANTHIES
9 letters: XANTINES
10 letters: XENELASIES

Table 1. Evil Motus Index (lines correspond to the length of the words and columns to the initial letter)

	A	**B**	**C**	**D**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
5	5	5	**7**	5	4	5	4	4	4	4	5	5	5	4	5	3	**6**	5	5	3	5	3	3	3	4
6	5	5	5	5	4	4	4	4	4	4	4	5	4	4	4	4	5	4	4	3	4	3	3	3	3
7	5	5	4	5	4	4	4	4	4	4	4	4	4	4	4	4	5	4	4	3	4	3	**2**	3	3
8	4	4	4	4	4	4	4	4	3	3	4	4	3	4	4	3	4	4	4	3	4	3	**2**	3	3
9	4	4	4	4	4	4	3	4	3	3	4	4	3	3	4	3	4	4	4	3	4	3	**2**	3	3
10	4	4	4	4	4	4	3	4	3	3	4	4	3	3	4	3	4	4	4	3	3	3	**2**	3	3

A second series of results can be put forward. These are related to 5-letter words beginning with C or R. Indeed, these have an EMI greater than 5, which is the maximum number of proposals in the rules for 5-letter words. As said previously, the EMI represents the worst case for any strategy. In other words:

Proposition 1. *There is no strategy to find all the 5-letter words with the number of tries allowed in the initial rules.*

4 Decision procedures

The Minimax algorithm, mentioned in the previous section, guarantees to find as many words as possible. It does not take into account the number of proposals needed and takes an utterly pessimistic view. Nevertheless, as mentioned in the introduction, in some variations of the game, it is better to guess the word in as few attempts as possible. For example, the player can be given a limited time to guess a predefined number of words. It is then not possible to use all the tries to guess each one of the words, but it is possible to fail on certain words, even if it means having to guess the following ones more quickly.

In this section, we look at other approaches in order to limit as much as possible the average number of tries used per word, without necessarily trying to guess as many words as possible. For that purpose, we consider $Comp(D_l^n, w, S)$ the set of words of D_l^n *compatible* with the set of proposals S when the word to guess is w (that is, all the words with the same correctly-placed and incorrectly-placed letters as the set of proposals S)[1]. Thus this is a game with complete information.

Given a set of proposals S, BP^S and MP^S the indices of the correctly-placed and incorrectly-placed letters, we associate with each word $w \in D_l^n \setminus S$, a vector of the sizes of the spaces of the possible words $\nu(w) = \langle \alpha_1, \alpha_2, ..., \alpha_m \rangle$ such that if $Poss(D_l^n, S, BP^S, MP^S) = \{w_1, w_2, ..., w_i, ..., w_m\}$:

$$\alpha_i = |Comp(D_l^n, w_i, S \cup \{w\})|$$

[1] Note that $Comp(D_l^n, w, S) = Poss(D_l^n, S, BP^S, MP^S)$ but the inputs are different as they depend on the available information.

i.e. for each word w which is valid and has not been proposed yet and for each possible word w', we calculate the number of compatible words when w is proposed next and w' is the actual word to guess.

This amounts to calculating the size of the solution space for each valid word that can still be chosen.

Example 4 *Consider the following dictionary:*
$D_L^7 = \{LEGALES, LIGNIEZ, LIGNINE, LIGNITE, LIGUIEZ, LIVIDES, LOGIONS, LOGIQUE\}$.

The player proposes "légales" first:

Then the player proposes the word "livides":

In this case, only the words "lignine" and "lignite" are possible (i.e. could be the word to guess) and the valid words not proposed yet are: "ligniez", "lignine", "lignite", "liguiez", "logions" and "logique". The following table presents, for each valid word not proposed yet (in the first column), the number of compatible words if it is proposed next, depending on the possible words (on the first line).

Table 2.

	LIGNINE	LIGNITE
LIGNIEZ	**2**	**2**
LIGNINE	1	1
LIGNITE	1	1
LIGUIEZ	2	2
LOGIONS	**1**	**1**
LOGIQUE	2	2

For example, the player considers proposing the word "ligniez" next[2]*. If the word to guess is "lignine", the environment will indicate that the first five letters are correctly placed, that the letter E is incorrectly placed and the letter Z absent. Then there will still remain two possible words. In the same way, if the word to guess is "lignite", there will also remain two possible words.*

Now if the player considers proposing the word "logions" next then there will remain only one possible word: the word to guess.

In a decision-making point of view, given a dictionary D_l^n and a set of proposals S, the set of the "best" words to choose is the set of preferred words for a total preorder $\trianglelefteq$ such that :

$$C = \{w \in D_l^n \setminus S : \forall \omega' \in D_l^n \setminus S, w \trianglelefteq w'\}$$

[2] The player can propose a word if she thinks it can provide useful information even if she knows for sure that this word cannot be the one to guess.

When this set is not reduced to a singleton, we consider as tie-break by default the alphabetical order.

4.1 The different aggregate and comparison functions used

Table 3.

	LIGNIEZ	LIGNINE	LIGNITE	LIGUIEZ	LOGIQUE	Sum	Max	Gini	Entropy
LIGNIEZ	1	2	2	1	1	7	2	0.17	2.24
LIGNINE	1	1	1	1	1	5	1	0.00	2.32
LIGNITE	1	1	1	1	1	5	1	0.00	2.32
LIGUIEZ	1	2	2	1	1	7	2	0.17	2.24
LIVIDES	2	2	2	2	1	9	2	0.09	2.28
LOGIONS	2	2	2	1	1	8	2	0.15	2.25
LOGIQUE	1	2	2	1	1	7	2	0.17	2.24

Consider table 3, built after the first proposal of the player (see Example 4). Once the table is built, it remains to determine which aggregate function to use for each valid word. The various aggregate functions which we implemented are the following ones.

Sum The *Sum* function selects a word whose sum of possible words is the lowest. In the case above, the words are associated with values from 5 to 9. The words "lignine" and "lignite" are the ones associated with the value 5. One of these words will thus be chosen.

Max The *Max* function selects a word whose maximal value is the lowest. In the example above, the words "ligniez", "liguiez", "livides", "logions" and "logique" are associated with the value 2, while the words "lignine" and "lignite" are associated with the value 1. One of these latter words will thus be chosen.

Gmax The *Gmax* function [6,4] selects the word whose maximal number of possible words is the lowest. In case of a tie, it selects the word whose maximal number of possible words, after removing one occurrence of the maximal number of possible words, is the lowest, and so on, until only one word has the lowest maximal number of possible words or until the minimal and the maximal numbers of possible words are equal.

Gini The function, based on the *Gini* index [2], an index used in economics in the context of the distribution of wealth, selects the word for which the number of possible words has the lowest Gini index, that is to say, whose distribution is the most "egalitarian". Let $v = \langle \alpha_1, \ldots, \alpha_n \rangle$ and σ be a permutation such that $\sigma(v) = \langle \alpha'_1, \ldots, \alpha'_n \rangle$ where $\forall i, j \in \{1, \ldots, n\}$, $i \leq j \iff \alpha'_i \leq \alpha'_j$ (i.e. $\sigma(v)$ is

the vector of the values of v, sorted in ascending order). The Gini index of v is defined as

$$G(v) = \frac{2\sum_{i=1}^{n} i\alpha'_i}{n\sum_{i=1}^{n} \alpha'_i} - \frac{n+1}{n}$$

Entropy The *Entropy* function selects the word whose entropy [7] is the lowest. Let $v = \langle \alpha_1, \ldots, \alpha_n \rangle$ and $v'\langle \alpha'_1, \ldots, \alpha'_n \rangle$ its normalized counterpart (i.e. $\alpha'_i = \alpha_i / \sum_{i=1}^{n} \alpha_i$), the entropy of v is defined as

$$H(v) = -\sum_{i=1}^{n} \alpha'_i \log(\alpha'_i)$$

Bobo The *Bobo* function is a control method which alphabetically orders the words and chooses the first possible word from the list. This is the first method we implemented in a Nao robot [1].

4.2 Possible Gmax

This function is a variant of Gmax where, instead of considering the set of valid words, we consider only the set of possible words. This variant gave good results only within the framework of Gmax.

4.3 Tie breaks

For each of the methods presented above, ties may occur. In this case, each of the algorithms chooses a possible word rather than a valid word that is not part of the possible words. If the tie is still not broken, by convention the first word in alphabetical order is chosen.

Thus, in Example 4, and for the aggregate function *Sum*, the words "lignine", "lignite" and "logions" are tied. The last word does not belong to the set of the possible words, consequently it is discarded. It is therefore the word "lignine", preceding "lignite" in the alphabetical order, which is selected.

SumGini The function *SumGini* behaves like the function *Sum* but, in case of tie break, it selects the word with the lowest Gini index.

5 Experiments

These methods were tested on the set of 5 to 10 letter words of the official Scrabble dictionary in the French version, in accordance with the standard rules described in Section 2. This corresponds to 212 017 words tested on the 9 methods in a cumulative time of more than 52 hours of calculation. For each word, we noted the number of attempts needed, including failures, i.e. when the granted number of tries was exceeded.

We then noticed there was a strong correlation between the length of a word and the number of tries needed to guess it. Some 5-letter words were impossible to guess, whereas all the 10-letter words were found by most of the implemented methods. We could also observe that, even if the Minimax algorithm guessed more words (except for the 5-letter words beginning with C or R) than the other methods, it also needed more attempts on average to guess a word. The results of the experiments are presented in Table 4. Averages are calculated including failures. The values concerning Minimax are given for information purposes, some values being missing (*timeouts*).

Table 4. Summary table

Method	**Average**	**Fails**	**Worst case**
Sum Gini	2,8145	53	8
Possible Gmax	2,8779	560	13
Sum	2,9834	81	8
Gmax	3,0117	75	7
Max	3,0482	76	7
Bobo	3,2852	1730	11
Gini	3,5434	2069	9
Entropy	3,7225	6596	10
Minimax	*3,3952*	*?*	*7*

If the average number of tries is considered as the key criterion, SumGini (the sum with the Gini index as tie-break) is the most effective before Possible GMax (cf. Figure 1). The method based on the Gini index gives results below the average. The Gini index makes it possible to highlight differences in terms of distribution, but it does not take into account the scale of this difference. Thus, the distribution 1, 1, 1, 1, 10 will have the same index as the distribution 1, 1, 1, 1, 100. Nevertheless, the Gini index seems to be a good metrics to decide between tied words within another method. For example, the "SumGini" method, which applies the "Sum" method and, in the event of a tie, selects the word with the lowest Gini index, achieved good results.

On the other hand, if the main criterion is the longest sequence before guessing the word (*worst case*), Gmax, Max and Minimax are better (cf. Figure 2). Entropy-based methods perform poorly, regardless of the criteria.

Finally, note that if the 5-letter words starting with C or R are removed, Minimax guesses all the possible words. But it was not possible to determine the number of words that can not be guessed in these subsets of words, since the calculation times turned out to be prohibitive. A new implementation of the algorithm as well as additional experiments are needed to know the total number of words that this method can guess.

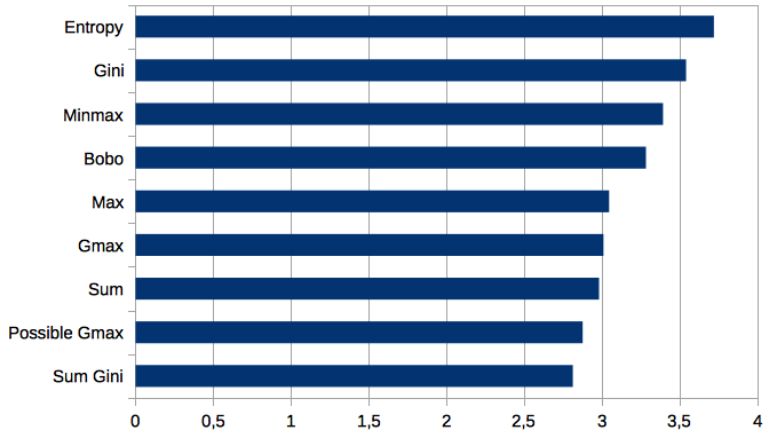

Figure 1. Average number of tries used

6 Conclusion

In this article, we presented a set of methods to play Motus effectively. We first demonstrated that for some common dictionaries it is not always possible to define a winning strategy. We also showed that even if the Minimax algorithm can guess most of the words, it requires using more tries on average than other methods. Thus the algorithm to use depends on the adopted rules. If the main criterion is to guess as many words as possible, regardless of the number of tries needed (as is the case, for example, in the first round of the television show), the Minimax algorithm must be favored with the exception, maybe, of 5-letter words beginning with the letters C or R. If, on the contrary, one tries to reduce as much as possible the average number of attempts to guess a word, even if it means failing from time to time, other methods are preferable.

The algorithms we developed are used as a demo for the general public[3]. People are invited to propose a list of words that, according to them, our algorithm will not be able to guess in a limited time. For example, the public proposes a list of 10 words, which the software has to guess within 3 minutes. If the software guesses all the words within the granted time, it wins the game. If it fails to guess a word or exceeds the allotted time, the audience wins. We impose here the two constraints mentioned above: it is both necessary to find all the words and to limit as much as possible the average number of tries used. For the software to maximize its chances of winning, it must therefore find a compromise. Finally an online version of the demo can be found here: `http://lagrue.ninja/motus/`.

It would be interesting to test the different algorithms on other dictionaries, in French but also in other languages. We also plan to weight the different possible words according to their probability of occurrence in the French language, rather

[3] `https://youtu.be/s-1ySiJR0ew`

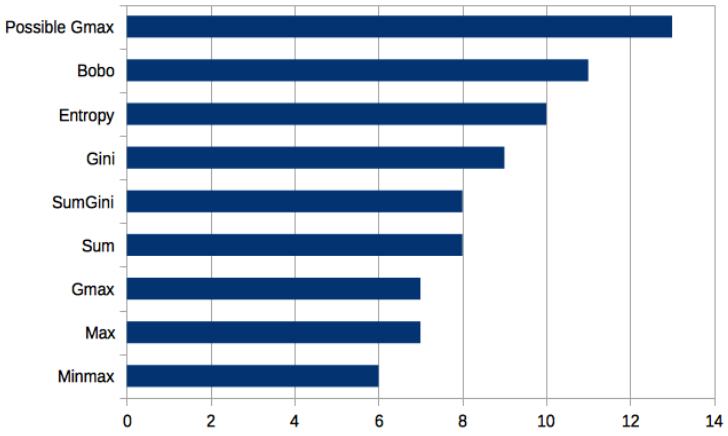

Figure 2. Worst tries

than to consider that they are equiprobable, then to observe the consequences of these choices on the results.

7 Acknowledgments

The authors would like to thank Francois Bonnet for his insightful comments and for presenting the original version of this paper at CG 2008.

References

1. Cardon, S., Delorme, F., Lagrue, S.: Nao joue à motus. In: actes du 18ème congrès francophone sur la Reconnaissance des Formes et l'Intelligence Artificielle (RFIA'12). p. 978 (2012), (Demo session)
2. Gini, C.: Measurement of inequality of income. Economic Journal 21, 22–43 (1921)
3. Knuth, D.E.: The computer as master mind. Journal of Recreational Mathematics 9(1), 1–6 (1976)
4. Konieczny, S., Pino Pérez, R.: Merging information under constraints: a logical framework. Journal of Logic and Computation 12(5), 773–808 (2002)
5. Kooi, B.: Yet another mastermind strategy. International Computer Games Association Journal 28(1), 13–20 (2005)
6. Moulin, H.: Axioms of Cooperative Decision Making. Cambridge University Press (1988)
7. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal 27, 379–423 (1948)
8. Von Neumann, J., Morgenstern, O.: Theory of Games and Economic Behavior. Prince-ton University Press (1947)