# Cheating is not playing: *Methodological Issues of Computational Game Theory*

**Bruno Beaufils**[1] and **Philippe Mathieu**[2]

**Abstract.** Computational Game Theory is a way to study and evaluate behaviors using game theory models, via agent-based computer simulations. One of the most known example of this approach is the famous Classical Iterated Prisoner's Dilemma (CIPD). It has been popularized by Axelrod in the beginning of the eighties and had led him to set up a successful Theory of Cooperation.

This use of simulations has always been a challenging application of computer science, and of agent-based approaches, in particular to Social Sciences. It may be viewed as *Empirical Game Theory*. These kind of approach is often necessary since, in the general case, classical analytical ones do not give suitable results. These tools are also often used when full game-theoretic analysis is intractable.

The usual method to evaluate behaviors consists in the collection of strategies, through open contests, and the confrontation of all of them as in a sport championship. Then it becomes, or at least seems to become, easy to evaluate and compare the efficiency of these behaviors.

Evaluating strategies can however not be done efficiently without the insurance that algorithms used are well formed and that they can not introduce bias in their computation. It can not be done without tools able to prevent or, at least, measure deviation from the object of the study. Unfortunately people using such simulations often do not take care seriously about all those aspects, because they are not aware of it, and sometimes because they are. We will try to show effects of bad simulations practice on the simplest example.

We show methodological issues which have to be taken care of, or avoided in order to prevent trouble in simulation results interpretation. Based on some simple illustration, we exhibit two kinds of bias that could be introduced. We classify them as voluntary or involuntary mistakes. The former ones can be explained by poor design of experimentations whereas the latter can defeat the purpose of the evaluation using simple ideas of agreement and cooperation. We also show the implications on interpretations and conclusions that such errors may produce.

We state that scoring/ranking methods are part of the game, and as such have to be described with the game. Many points described may seem to be widely known. We think that with the growth of interest of such methods they have to be detailed and exposed clearly.

## 1 Introduction

Social sciences generally deal with the study of human beings behavior and their effect on the behavior of groups of humans beings. Studies in this field use two methodologies. The first one consists, as in *natural science*, in the observation of real life situations in order to try to determine which laws govern the observed groups. The second one, whose adoption is much more recent, consists in the construction of formal models which are then tested and which results are compared to observations. Actually, those two methodologies are not as distinct as one may think. Often the first method is used as first step for the second one in a infinite loop trying to understand the world.

A lot of scientific fields are using this kind of methods to achieve their goal. One may think to economic and management science, trying to understand the behavior of group of people, psychology and artificial intelligence, trying to understand or mimic the behavior of individual. Some mathematical tool have been set up with such a goal in mind. Game Theory has, for instance, been mainly set up in order to understand and explain the behavior of people in parlor game, [8]. It has been used in many political situation, trying to solve conflicts (as this year Economics Nobel Prize winner work) and is still very largely used to understand economic behavior.

With the growth of computer power the testing steps of these approaches are more frequent. The main reason is that purely mathematical tools are sometimes nor powerful nor usable enough when dealing with individual elements and not sets. These tools are very well adapted for a macroscopic view of the world but lose a lot of attractiveness when trying to have a view at a microscopic level. These inappropriate perspectives may be caused by the classical continuous approach of mathematical tool used to explain a world which is a discrete one. On the other side, computers and agent-based models deal very well with discrete computations. Combination of discrete mathematical models and computation power of today is a real chance for *social science* at large.

Another explanation of this proliferation of computer simulations use may be that people think that computer are easier to use than mathematics. The problem is that it is generally not true. We will try to show difficulties involved when using computer agent-based simulations along with some basic solutions which are usable.

We will focus on one particular example which is the study of individual behavior and their effect mainly on the individual fitness but also on the group behavior. This is exactly what computational game theory or *empirical game theory ([9])*, is about. While classical game theory try to solve (or at least to explain) conflictual situations and evolutionary game theory try to explain population dynamics, computational game theory is about evaluating individual behavior. Within this context the purpose of the work is then to find some good behavior, also called strategies. What *good* really means depends on the macroscopic effect studied (cooperation, coalition dynamic, etc.).

The classical way to do is to simulate a big tournament implying a lot of different behavior in a specific game. The more efficient the

[1] LIFL, University of Sciences and Technologies of Lille, beaufils@lifl.fr
[2] LIFL, University of Sciences and Technologies of Lille, mathieu@lifl.fr

behavior is in this game the better it is considered. Diversity in nature of behavior present in the population of agents implied is then very important. Evaluating an agent in a population where every individuals behave the same way is not interesting, at least most of the time. In order to ensure this needed diversity scientists often use a way which seems efficient : ask other scientists how they will behave in the specified game. That has been successful more than once. One of the main reason is that every participant has not the same representation of the situation modeled by the game. Since background of participants are different, they even do not have the same idea of how to deal well in such situation.

Evaluating strategies can not be done efficiently without testing it more than once. It can not be done without the insurance that algorithms used are well formed and that they can not introduce bias in the result of simulations. It can not be done without tools able to prevent, or at least to measure, deviation from the object of the study.

Unfortunately people using computer simulations does not always take care seriously about all these aspects (CEC'04 contest), because they are not aware of it, and sometimes because they are. We will try to show effects of bad simulations practice on the simplest example.

In a first section we will describe computational game theory methods which will be used to illustrate our ideas. A specific example will be chosen, described, and used in all the rest of the paper. The next section will describe some bias that could be introduced by those methods when they are not used in a proper manner or simply because they are not well designed. That could be summarized as *how to cheat without communication*. Then the third section will described how strategies involved can defeat the purpose of the evaluation using trivial ideas based on agreement and cooperation. That will be summarized as *how to cheat with communication*.

## 2 Computational Game Theory

Game Theory may be seen as a mathematical tool designed to understand what is a rational behavior in specific situations. More precisely behavior are called strategies. Strategies describe in detail what to do in any possible situation. Possible situations are described by some rules. A definitive definition of what Game Theory is about is still very discussed and two different approaches still coexist. Even the *real life* applicability of Game Theory is still widely discussed. One can refer, among others, to [3, chapter 1] or [7, chapter 5].

We would like to show that evaluating strategies in game theory by computer simulations has to be done with scientific rigor and needs some methodological precautions. One has also to be aware of limits implied by those issues. We focus the demonstration on iterated and repeated games involving two players. The main used evaluation method is based on sport championship. The idea is to evaluate and then order some strategies in a specific game. Evaluation is done pairwise, for every possible pair. In this paper we concentrate, as an example, on the iterated prisoner's dilemma, see [6] for complete history and informal presentation. Remarks made in this context stay however true on any iterated game.

Classically, these kind of evaluations are done during a specific contest where any kind of people are allowed to submit strategies, through a simple description or a more detailed source code implementation in a fixed programming language. The main benefit of using such situation is clarity, since distinction between people being evaluated (*participants*) and people evaluating (*referees*) are obvious.

Two cases are then observable: (i) implementation mistake or design flaws (ii) organized cheating. In both cases we will show that it is almost always possible to get a well-evaluated strategy at the end of the contest although it is actually a very poor one. In the first case it is often due to a naive software implementation implying edge effect. Referee, and not participants, has then to be blamed. It could be defined as *involuntary cheating*. In the second case an unfair participant is able to profit, for instance, of a design flaw in the contest organization, in order to favor a strategy it has submitted. It could be defined as *voluntary cheating*.

### 2.1 Iterated games

Iterated games studied here are based on the repetition of a simple game which is defined by a single payoff matrix.

Every iteration of a game is defined by choices made by players. Choices consist of a selection in a fixed list of available moves[3]. Combination of the move selected by players in a specific iteration defines an outcome of the simple game. The payoff matrix defines payoff obtained by each player for any reachable outcome in an iteration. Iteration will be later called round; the complete sequence of iterations referenced as a meeting, a match or simply a game. The final payoff of a player is simply the sum of all iteration's payoff. The goal of all players is the same: to maximize their final payoff.

A player is identified by predetermined choices following a strategy. This strategy allows, for any given iteration, to know the move a player will use according to previous reached outcomes and in particular according to previously moves played by opponents.

Using computer science terminology a strategy is identified to a program and a behavior during an iterate game to an execution of this program.

In the CIPD there are only two players and two moves available in the simple game. These moves are noted C (for Cooperation) and D (for Defection). For every iteration players choose their moves simultaneously. The payoff matrix is presented in table 1. This matrix is known by both players.

**Table 1.** Classical Iterated Prisoner's Dilemma payoff matrix.

| | C | D |
|---|---|---|
| C | 3 | 0 |
| D | 5 | 1 |

*This matrix is a Prisoner's Dilemma payoff matrix. This game involves 2 players and 2 different moves (C and D). Available moves are the same for each player. It is a symmetric matrix: payoff of player A for outcome (C,D), that is when A played C and B played D, is the same as payoff for player B for the outcome (D,C), that is when A played D and B played C; which is 0. Matrix shows thus only the payoff of row player.*

In this paper we will only use simple strategies usable in the CIPD:

`all_c` always plays `C` whatever opponent played.
`all_d` always plays `D` whatever opponent played.
`tit_for_tat` plays `C` as first move, then plays whatever move opponent played in the previous iteration.
`spiteful` plays `C` until the opponent played `D`, then always `D`.
`soft_majo` plays the move the opponent has used more often in previous iterations (the majority move). If it played `C` as many time as `D` then plays `C`.
`per_X` plays periodically moves in the sequence X. For instance `per_CDD`, also noted as $(\texttt{CDD})*$, plays periodically `C`,`D`,`D`.

[3] These moves are the available strategies of the simple game

## 2.2 Evaluation method

The most used strategy's evaluation method is simply the comparison of its payoff to the one of another strategy. The higher the payoff is, the better the strategy is considered. To evaluate all strategies of a fixed set it is then sufficient to sum scores of each strategies against every other and finally to rank them on the basis of their scores.

For a two-player game, computing such a *round-robin* tournament on set $\mathcal{S}$ containing $n$ strategies is simply to fill a $n \times n$ matrix and then to sum value of each lines in order to get the score of each strategy.

It has to be noticed that in such tournament a strategy's score does not necessarily include its score when playing against itself. To the end of the paper and in order not to deal explicitly with the contribution of such inclusion, at least in term of robustness, we will consider that all cells on the diagonal of tournament matrices are nulled. That simplification does not change anything to ideas developed herein and almost nothing to examples chosen as illustrations.

With such a method the higher a strategy is ranked the better its value is considered. Quality of the evaluation depends however essentially in the size of the strategies set.

Classically in order to get a big number of strategies one may organize a contest open to everyone. It is asked to every participant to submit **one** strategy which respects rules of the played game. A big tournament is then computed with all submitted strategies as the set to be evaluated. The basic idea is here to allow the research of new *good* strategies for a specific game. The contest is used as a way to insure diversity and heterogeneity in behaviors.

This method has been regularly used, in particular on the CIPD. One can think to Robert Axelrod contest in the beginning of eighties, see [1], to the French edition of the Scientific American in the beginning of nineties, see [4] for the announcement and [5] for the results, as well as to some scientific conferences such as Artificial Life V in 1996 or the Conference on Evolutionary Computation in 2000. In scientific conferences the goal is often to test new techniques (such as evolutionary computation) when applied to strategies for the iterated prisoner's dilemma.

In all cases *modus operandi* is always the same. Someone asks people to submit strategies. Let us call it the organizer. It has the responsibility to collect strategies and to test them in the most objective way using computer simulations. People submitting strategies are called participants. They submit their proposition by sending the organizer some source code of a computer program, a textual description, or simply by filling a web form, explaining the behavior they want to propose. The organizer generally is in charge of objectivity of the contest. It is the one who runs simulations and thoroughly study their results. The only purpose of such contest is to determine some efficient strategies in a specific situation, not to find a *realistic* mathematical model of a social interactive situation. The organizer insures that all participants are aware of that fact.

# 3 Cheating without communicating

## 3.1 Normalization

One of the well known problems with iterated version of prisoner's dilemma is the length of games. If both players know when the meeting will end (last iteration) then an iterated game is equivalent to a simple one. In such situation both players have to play `D` all the time. It has to be avoided.

One of the chosen solution is to use a discount rate. It allows to give less signification to future moves compared to already played one. For any formal studies with infinite meeting without any real or computed simulations this make sense. Outside of this context it becomes very difficult to chose a value for this rate and even more difficult to justify this choice. Moreover infinite is not a realizable concept in computer science.

When using simulations it is often preferred to pseudo randomly choose a length at the beginning of every game of a tournament. This pseudo-random number generation is done for a fixed variance. Result of the generation is of course not available to players nor before, nor during the meeting.

It may however be possible for an unfortunate player to always get short meeting compared to the average length. In such conditions it may be badly evaluated whereas, with some luck, evaluation could be completely different. Let us illustrate this using 3 strategies `all_d`, `tit_for_tat`, `(CD)`$*$, and consider the following game length:

| Players | Game Length |
|---|---|
| `(CD)`$*$ vs `all_d` | $m$ |
| `all_d` vs `tit_for_tat` | $n$ |
| `tit_for_tat` vs `(CD)`$*$ | $p$ |

According to these length, to the CIPD payoff matrix (table 1) and the definition of strategies, payoff obtained by each strategies during a tournament are:

| | `all_d` | `(CD)`$*$ | `tit_for_tat` |
|---|---|---|---|
| `all_d` | | $6(m/2)$ | $n+4$ |
| `(CD)`$*$ | $m/2$ | | $(5p/2)+3$ |
| `tit_for_tat` | $n-1$ | $(5p/2)-2$ | |

With $n = 50$, $m = 100$, $p = 60$, which means an average deviation of 20, the obtained rank is:

1. `all_d` with 354 points ;
2. `(CD)`$*$ with 203 points ;
3. `tit_for_tat` with 197 points.

On the other hand with $n = 50$, $m = 60$, $p = 100$, which means we stay at a average deviation of 20, the rank becomes:

1. `tit_for_tat` with 297 points ;
2. `(CD)`$*$ with 283 points ;
3. `all_d` with 234 points.

It is important to notice that these 2 ranks have completed different head and tail. It is also to be noticed that average deviation and standard deviation stay constant in both cases: 20 and 26.46.

With a normalization of game length to $n = m = p$, we then get the following rank:

1. `all_d` with $4(n+1)$ points ;
2. `(CD)`$*$ with $3(n+1)$ points ;
3. `tit_for_tat` with $\frac{7n}{2} - 3$ points.

Naturally for the two previous tournament using another normalization process, consisting in the computation of the average score for each iteration, we get a similar rank, which is however different from the two firsts:

1. `all_d` with an average payoff of 4 points ;
2. `tit_for_tat` with an average payoff of 3 points ;
3. `(CD)`$*$ with an average payoff of 3.5 points.

Game length has to be unknown by players. Contest organizer has however to choose between using the same pseudo-randomly computed length for every game in the contest, or normalizing scores according to number of rounds played by each strategies. If a tournament is used as an evaluation method player do not have to play against itself.

In the rest of this paper we will use game of 10 rounds for every example and illustration.

### 3.2 Clones strategies

In some particular context 2 different strategies (program) may produce the same behavior (execution). Let us consider for illustration `tit_for_tat` and `spiteful`. They are different since they use different concepts. They however produce the same behavior against 2 strategies as uncommon as `all_c` and `all_d`.

In a general manner the same case may arise when dealing with genotype and phenotype. Using computer science terminology it is trivial to say that two completely different programs may produce the same results.

That may be viewed from another angle. Many strategies with different names may finally be exactly the same ones. In such case, the use of such a strategy in a contest is a clear bias of the final evaluation and thus devaluate the quality of it.

If, for instance, we use $n$ times the same strategy `S1` (by means of same code but different names) in a set of strategies all different when compared by pair (by mean of different code), then it may be possible that `S1` is the best evaluated one. When the same strategy is used only once with the same set of competitors then it may be another strategy than `S1` which is considered the best.

Considering `tit_for_tat` as `S1` as well as `all_d`, and `(CD)`$*$ for the set of competitors it is then easy to see that such situation may arise very clearly, even with tournament composed by game of only 10 rounds:

```
TOURNAMENT RANK          TOURNAMENT RANK
1 :       all_d = 44     1 : tit_for_tat = 62
2 :      per_cd = 33     2 : tit_for_tat = 62
3 : tit_for_tat = 32     3 :      per_cd = 61
                         4 :       all_d = 58
```

We showed how the repeated use of a strategy may influence results of tournament. This frequent bias may arise in 2 cases:

- when 2 strategies have similar implementation code but different names. It is then very easy for a participant of a contest to favor its proposition by flooding it with a very big number of the same strategy using different names each time. In another context (combinatorial auctions, see [10]) a problem close to this one has been identified as *false-name bid*.
- when 2 strategies have different code, different names but produce the same execution due to the simulation context. It is a classical problem in computer science since there is an infinite number of way to write the same thing.

Moreover, it has to be noticed that deciding if two strategies are similar in the general case is impossible. That is a very strong knowledge in theoretical computer science derived from the fact that it is impossible to decide if two programs (Turing machines) have identical behavior (accept the same language). This decision problem is not recursively enumerable. It is however perfectly decidable in some more constraints cases where strategies are defined by filling a predefined structure. This last method is often used on web contest where participants fill web form in order to define a strategy. This is, however, less interesting since, in such cases, it is often possible to make some tournament involving all possible instances, see work presented in [2] as an example.

## 4 Cheating with communication

In most cases, players do not know strategies used by their opponents. They may try to deduce it from the behavior of others players. This is a difficulty and one of the major interest of using computational game theory, which we can call *behavioral inference*. At the opposite, let us suppose that you know the strategy of your opponent before playing. In this case it could be easy to determine the best reply to it and thus to maximize your payoff.

Although it is forbidden to communicate *directly* with its opponent, in order for instance to make a deal and agree on the behavior to adopt, it is possible to use the history of played moves to communicate *indirectly*. One of the most trivial idea is then to use some kind of starter succession of moves, used by others to identify you.

Communication can then be established by some coding system fixed in advance by the strategy creator.

With such hypothesis two kind of cases may be considered:

- a unique strategy used by different participants who try to recognize themselves;
- different strategies whose goal of some is to favor the profit of some other.

### 4.1 Strategies which recognize themselves

We illustrate one such case through a strategy which plays a specific starter succession of moves and then cooperates always after this succession if it observed the same starter played by the opponent and else defects.

If half the population uses the same strategy, then one member of the subpopulation using it has a great chance to be the winner. On the other hand if only one participant uses it in the population then it has a great chance of being a loser since it could no more take advantage from the communication.

Let us consider the `cheater` strategy which plays the `(CDDC)` starter, then observe what the opponent played in the 4 first moves. If opponent played the same succession of moves as the starter then `cheater` cooperates always, else it defects always.

If 3 players using `cheater` are tested against `tit_for_tat`, `spiteful` and `soft_majo` the tournament winner is `cheater`. If the same experiment is done with only one instance of `cheater` then it appears to be the last of the final tournament rank:

```
TOURNAMENT RANK          TOURNAMENT RANK
1 :    cheater2 = 109    1 :    spiteful = 75
2 :    cheater3 = 109    2 : tit_for_tat = 74
3 :     cheater = 109    3 :   soft_majo = 73
4 :    spiteful = 105    4 :     cheater = 57
5 : tit_for_tat = 102
6 :   soft_majo = 99
```

Once 2 players using `cheater` recognizes themselves they win 3 points each and on each round, whereas the rest of time they make others players losing at least 2 points.

Generally this kind of behavior is easily identifiable on tournament results since all strategy using it are grouped at the same level in the final rank (at least when using purely deterministic strategies). This

does not show the same effect as in the previous section where the same strategy is used by more than one player. Here strategies have different behavior adapted to different opponent.

### 4.2 Master/Slaves

On the same principle as in the previous example it is possible to establish a strategy taking advantage of some others agreeing strategies. The one who benefits is said to be the master and the others to be the slaves.

Compared to the previous example once strategies have recognize themselves then on each round the master win 5 points defecting against its slaves, which agree to be slave by cooperating only with the master. Against others players slaves may use any other aggressive behavior.

Let us consider that the `master` strategy plays (`CDDC`), then if it recognized the same starter played by its opponent always plays `D`, and if not plays as `tit_for_tat`.

The slave plays exactly as `cheater`: it plays (`CDDC`) then if the opponent did the same starter it cooperates, and else defects.

If `master` is evaluated against `cheater`, `tit_for_tat` and `soft_majo` it wins the tournament whereas the slave one is ranked last. In the case of no slave the `master` takes the last position:

```
TOURNAMENT RANK          TOURNAMENT RANK
1 :      master = 105    1 : tit_for_tat = 84
2 : tit_for_tat = 98     2 :   soft_majo = 83
3 :   soft_majo = 96     3 :    spiteful = 75
4 :    spiteful = 90     4 :      master = 67
5 :     cheater = 65
```

Such example could even be refined considering one master but $n$ slaves with completely different starter for each slave. The master job would then only be to recognize the different slaves and to use some other rather *good* strategies against *real* opponent such as `tit_for_tat` or `spiteful`.

It has to be noticed that the harmful effect of slaves can be selective. For instance a slight modification of `cheater` (replacing `all_d` by `tit_for_tat`) may harm only aggressive strategies.

In a real contest one has however to find another participant who accepts to be slave, that is using a strategy which have a lot of chance to be badly evaluated. If such coalitions could be identified one solution is then to set the score of each member of the coalition as the average payoff of all member of it. Unfortunately this identification is often difficult to do

Public contest may avoid as far as possible the possibility for one participant to offer more than one strategy. Collusion, such as those shown here, may nevertheless appear since agreement or coalition between different participants before the contest are not avoidable.

The final rank has then to be particularly well studied in all its details, and, at least, strategy of players with the same final payoff have to be investigated.

## 5 Conclusion

*Computational Game Theory* deals with the evaluation of heuristic strategies using iterated game. It is some kind of *empirical* Game Theory. When full game-theoretic analysis is intractable or unuseful, and, as in the general case, it is not possible to prove formally that a strategy is better than another, computer simulations become essential. It generally uses a very specific multi-agent based approach where agent's behaviors are defined by human participants in public contest. All around the world this method is very often used on different kind of game.

Through the CIPD, we showed in this paper that this kind of evaluation can not be done without any precaution, and that false, or at least biased, results may be found. It is very easy to favor one particular kind of strategy, consciously or not. We described different types of skews one should be aware of and take care about (computation method, repetition of strategies, master/slave effects, etc.).

All along the presentation we tried to give number of examples which we described precisely so that they can be verified and reproduced. We tried to make a short survey of what has to be done or not when using computer simulations for iterated game strategies comparison. Even if you could not present them here, methodological solutions to problem presented here exist. They are mainly based on evolutionary computation ideas and on sets of reference.

In this particular context of strategies evaluation, we basically state that scoring/ranking methods are part of the game. They thus have to be described with the game. All participants must have the same level of information on the contest in order, for instance, to arm themselves against some flaws described here. Changing the purpose of a contest after its start or favoring some participants is cheating.

Many points described here may seem to be widely known or at least trivial. With the rise of contests falling in all traps described here, we fear that finally they are not so well known. We think that with the growth of interest of such methods they have to be detailed and at least exposed clearly.

Computational, as Empirical Game Theory are crossing a major step of its evolution. It may be a very efficient method for evaluating and later constructing new *artificial intelligent* behaviors. In order to achieve this goal, it has to be used seriously.

## 6 Acknowledgements

Tools used to made experiments presented here are all available on the web at the following address:`http://www.lifl.fr/IPD`.

Works presented here has been supported by the Nord/Pas-de-Calais regional council (through CPER TAC project) and the European Regional Development Fund. Authors would like to thank all of theses institution for their support.

## REFERENCES

[1] R. Axelrod, *The evolution of cooperation*, Basic Books, New-York, USA, 1984.
[2] B. Beaufils, J.-P. Delahaye, and P. Mathieu, 'Complete classes of strategies for the classical iterated prisoner's dilemma', volume 1447 of *Lecture Notes in Computer Science*, 33–41, Springer-Verlag, (1998).
[3] K. Binmore, *Essays on the Foundations of Game Theory*, Blackwell, 1990.
[4] J.-P. Delahaye, 'L'altruisme récompensé ?', *Pour La Science (French Edition of Scientific American)*, **181**, 150–156, (1992).
[5] J.-P. Delahaye and P. Mathieu, 'L'altruisme perfectionné', *Pour La Science (French Edition of Scientific American)*, **187**, 102–107, (1993).
[6] W. Poundstone, *Prisoner's Dilemma*, Doubleday, 1992.
[7] A. Rubinstein, *Economics and Language*, Cambridge University Press, 2000.
[8] J. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944. The standard reference is the revised edition of 1947.
[9] W. E. Walsh, R. Das, G. Tesauro, and J. O. Kephart, 'Analyzing complex strategic interactions in multi-agent games', in *AAAI-02 Workshop on Game Theoretic and Decision Theoretic Agents*, (2002).
[10] M. Yokoo, Y. Sakurai, and S. Matsubara, 'The effect of false-name bids in combinatorial auctions: new fraud in internet auctions', *Games and Economic Behavior*, **46**, 174–188, (2004).