On the Complexity of Game Isomorphism

Joaquim Gabarró and Alina García and M. Serna

2007
On the Complexity of Game Isomorphism
(Extended Abstract)*

Joaquim Gabarró  Alina García  Maria Serna

LSI Dept. Universitat Politècnica de Catalunya, Barcelona.
{gabarro,agarcia,mjserna}@lsi.upc.edu

Abstract. We consider the question of when two games are equivalent and the computational complexity of deciding such a property for strategic games. We introduce three types of isomorphisms depending on which structure of the game is preserved: strict, weak, and local. We show that the computational complexity of the game isomorphism problem depends on the level of succinctness of the description of the input games but it is independent of the way the isomorphism is defined. Utilities or preferences in games can be represented by Turing machines (general form) or tables (explicit form). When the games are given in general form, we show that the game isomorphism problem is equivalent to the circuit isomorphism problem. When the games are given in explicit form, we show that the game isomorphism problem is equivalent to the graph isomorphism problem.

Keywords. Game isomorphism, succinct representations, boolean formulas, computational complexity, boolean isomorphism, graph isomorphism.

1 Introduction

We are interested in the computational aspects of game equivalence. Surprisingly there is not a fully accepted definition of game equivalence in game theory books. In 1951, J. Nash [11] gave a definition of automorphism between strategic games. More recently, B. Peleg, J. Rosemuller and P. Sudhölter [13, 16] consider isomorphisms for strategic and extensive games with incomplete information. J.C Harsanyi and R. Selten have introduced other definitions of isomorphism [8]. Equivalence by the way of transformations to a common form have been considered in [4].

We consider the case of strategic games. Our motivation is twofold. First, a strategic game is a special type of combinatorial structure and a natural question is to study the computational effort needed to decide when two such structures are isomorphic. Secondly, we can ask if such a question is interesting to the game theory community. In practice strategic games are used as ingredient of more complicated games, but usually there is a way to transform any game into a strategic game. Furthermore, in [4] equivalence between extensive games is defined in terms of strategic games. Therefore strategic games are the first game structure to start analyzing game equivalence.

* Partially supported by Network of Excellence CoreGRID (IST-2002-004265), FET Proactive IP 001907 (DELIS) and by the spanish project TIN2005-09198-C02-02 (ASCE). The second author is supported by a FPI Spanish grant.
In defining a concrete equivalence we have to pay attention to the structural properties that are preserved in equivalent games. We consider three versions of isomorphisms. A strong isomorphism preserves utilities corresponding to the notion introduced in [11]. A weak isomorphism preserves preferences. A local isomorphism preserves preferences defined only on “close” strategy profiles. Each of them requires to preserve less information about the relative structure of profiles while preserving part of the structure of the Nash equilibria. More precisely, strong isomorphisms preserve pure and mixed Nash equilibria, while weak and local isomorphisms only preserve pure Nash equilibria.

In this paper we are interested in the computational complexity of deciding whether two games are equivalent. We consider four problems related to isomorphisms. In the IsIso problem, given two games $\Gamma$ and $\Gamma'$ and a mapping $\psi$ we have to decide whether $\psi$ is an isomorphism. In the IsAuto problem, given $\Gamma$ and $\psi$ we have to decide whether $\psi$ is an automorphism. In the Iso problem we decide whether two games are isomorphic. Finally in Auto we ask to decide when a game has an automorphism different from the identity. In order to study the computational aspects of problems on strategic games and isomorphisms, we need first to decide how to represent games and morphisms as inputs.

For games we consider the two levels of succinctness proposed in [2]. When a game is given in general form the actions are listed explicitly but utilities and mappings are given by deterministic Turing machines. In the explicit case utilities are stored in tables. In both cases morphisms are always represented by tables. This is not a restriction as in polynomial time we can pass a representation by Turing machines into a representation by tables. The main results of the paper are

- The IsIso and the IsAuto problems are $\text{coNP}$-complete, for games given in general form, and $\text{NC}$ when games are given in explicit form.
- The Iso and the Auto problems belong to $\Sigma^p_2$, for games given in general form, and to $\text{NP}$ when games are given in explicit form.
- The Iso problem is equivalent to boolean circuit isomorphism, for games in general form, and to graph isomorphism, for games given in explicit form.

The above results hold independently of the type of isomorphism considered, observe that boolean circuit isomorphism is believed not to be $\Sigma^p_2$-hard [1], as well as graph isomorphism is conjectured not to be $\text{NP}$-hard [9]. Therefore the same result applies for the Iso.

In our presentation we consider games defined by utilities, preferences, or local preferences to present our complexity results as general as possible and matching the corresponding definition of morphism. However a game defined by utilities can be defined by preferences or local preferences. All the results hold for games defined through utilities when the isomorphism is required to preserve utilities (strong), the induced preference relation (weak) or the induced local preferences (local).

The paper is organized as follows. In Section 2, we consider games defined through utilities and define the notion of strong morphisms. In this perspective we introduce the different problems on game isomorphism and formalize the
different levels of succinctness for the representation of games and morphisms. In Section 3 we develop the complexity results for strong isomorphisms. In Section 4 we study the complexity results for strategic games defined through preferences in the case of weak isomorphisms and we introduce the ideal local preferences (well adapted to the notion of Nash equilibrium). In Section 5 we study the structure of the Nash equilibria in the case of two player games, each player with two actions using weak isomorphisms. We conclude in Section 6, with additional comments and open problems.

Due to the lack of space all the proofs are omitted we refer the interested reader to the full version [6] for details. However, before formulating a hardness result we provide a definition of the transformation used in the reduction.

2 Games, Isomorphisms, Problems, and Representations

We start stating the mathematical definition of strategic games given in [12].

**Definition 1.** A strategic game \( \Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N}) \) is a tuple. The set of players is \( N = \{1, \ldots, n\} \). Player \( i \in N \) has a finite set of actions \( A_i \), we note \( a_i \) any action belonging to \( A_i \). The elements \( a = (a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n \) are the strategy profiles. The utility (or payoff) function \( u_i \) for each player \( i \in N \) is a mapping from \( A_1 \times \cdots \times A_n \) to the rationals.

First of all we consider game mappings which do not consider utilities. We adapt notations and definitions given in [13, 16].

**Definition 2.** Given \( \Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N}) \) and \( \Gamma' = (N, (A'_i)_{i \in N}, (u'_i)_{i \in N}) \), a game mapping \( \psi \) from \( \Gamma \) to \( \Gamma' \) is a tuple \( \psi = (\pi, (\varphi_i)_{i \in N}) \) where \( \pi \) is a bijection from \( N \) to \( N \), the player’s bijection, and for any \( i \in N \), \( \varphi_i \) is a bijection from \( A_i \) to \( A'_i \), the \( i \)-th actions bijection.

Observe that the player bijection identifies player \( i \in N \) with player \( \pi(i) \) and the corresponding actions bijection \( \varphi_i \) maps the set of actions of player \( i \) to the set of actions of player \( \pi(i) \). A game mapping \( \psi \) from \( \Gamma \) to \( \Gamma' \) induces, in a natural way, a function from \( A_1 \times \cdots \times A_n \) to \( A'_1 \times \cdots \times A'_n \), where strategy profile \( (a_1, \ldots, a_n) \) is mapped into the strategy profile \( (a'_1, \ldots, a'_n) \). A mixed strategy profile \( p = (p_1, \ldots, p_n) \) is given by probabilities \( p_i \) on \( A_i \) (such that \( \sum_{a_i \in A_i} p_i(a_i) = 1 \)) for \( 1 \leq i \leq n \). A game mapping \( \psi \) also induces a mapping \( \psi(p_1, \ldots, p_n) = (p'_1, \ldots, p'_n) \) such that \( p'_{\pi(i)} \) is a probability on \( A'_{\pi(i)} \) defined by \( p_{\pi(i)}(\varphi_i(a_i)) = p_i(a_i) \). Isomorphisms are game mappings fulfilling some extra restrictions about utilities or preferences. We start defining the stronger version of an isomorphism introduced by J. Nash [11], look also [13, 16].

**Definition 3.** Given \( \Gamma = (N, (A_i)_{i \in N}, (u_i)_{i \in N}) \) and \( \Gamma' = (N, (A'_i)_{i \in N}, (u'_i)_{i \in N}) \), a game mapping \( \psi : \Gamma \rightarrow \Gamma' \) with \( \psi = (\pi, (\varphi_i)_{i \in N}) \) is called a strong isomorphism \( \psi : \Gamma \rightarrow \Gamma' \) when, for any player \( 1 \leq i \leq n \) and any strategy profile
The strong isomorphism $\psi : \Gamma \rightarrow \Gamma'$ is $\psi = (\pi, \varphi_1, \varphi_2)$ where $\pi = (1 \rightarrow 2, 2 \rightarrow 1)$, $\varphi_1 = (t \rightarrow l', b \rightarrow r')$ and $\varphi_2 = (l \rightarrow b', r \rightarrow t')$. This strong isomorphism maps strategy profiles as $\psi(t, l) = (b', l')$, $\psi(t, r) = (t', l')$, $\psi(b, l) = (b', r')$ and $\psi(b, r) = (t', r')$.

The strong automorphism is $\psi' = (\pi', \varphi'_1, \varphi'_2)$ where $\pi' = (1 \rightarrow 2, 2 \rightarrow 1)$ and the action bijections are $\varphi'_1 = (t \rightarrow r, b \rightarrow l)$ and $\varphi'_2 = (l \rightarrow t, r \rightarrow b)$.

Fig. 1. Example of a strong isomorphism $\psi$ and of an automorphism $\psi'$.
\( \langle a, i \rangle \) is the value of the pay-off function of the \( i \)-th player on input \( a \). From the three levels of succinctness in the representations introduced in [2] we consider only two.

**Strategic games in general form.** A game is given by a tuple
\[
\Gamma = \langle 1^n, A_1, \ldots, A_n, M, 1^i \rangle.
\]
It has \( n \) players, and for each player \( i \), where \( 1 \leq i \leq n \), their set of actions \( A_i \) is given by listing all its elements. The description of their pay-off functions is given by \( \langle M, 1^i \rangle \).

**Strategic games in explicit form.** A game is given by a tuple
\[
\Gamma = \langle 1^n, A_1, \ldots, A_n, T \rangle,
\]
where \( T \) is a table such that \( u_i(a) = T[a][i] \).

In Definition 1 a game \( \Gamma \) is defined in an abstract way using set theory. When the computational aspects of \( \Gamma \) have to be studied, \( \Gamma \) has to be encoded. Here we consider two encodings with different levels of succinctness, the general form and the explicit form. We use the same symbol \( \Gamma \) to denote both, the abstract game and the encoded version. Observe that in the explicit form, games are described as it is done in elementary books, just giving explicitly the tables of utility functions. In order to describe a game mapping, we consider the less succinct approach.

**Game mapping in explicit form.** All is given explicitly, actions are given listing all its elements and mappings are given by tables, then
\[
\psi = \langle 1^n, A_1, \ldots, A_n, A_1', \ldots, A_n', T_\pi, T_{\varphi_1}, \ldots, T_{\varphi_n} \rangle
\]
where \( T_\pi, T_{\varphi_1}, \ldots, T_{\varphi_n} \) are tables such that \( T_{\varphi_i}[a_i] = a'_iT_\pi[i] \).

We have not considered the description of a mapping by Turing machines,
\[
\psi = \langle 1^n, A_1, \ldots, A_n, A_1', \ldots, A_n', M_\pi, M_{\varphi_1}, \ldots, M_{\varphi_n}, 1^i \rangle
\]
because in such a case we can construct an explicit coding of \( \psi \) with size bounded by \( 2|\psi|^2 \).

### 3 Complexity Results for Strong Isomorphism

First we consider the \text{IsIso} and \text{IsAuto} problems and later on the \text{Iso} problem. Our \text{coNP} hardness results follow from reductions from the following \text{coNP}-complete problem [7]:

**Validity problem** (\text{VALIDITY}): Given a boolean formula \( F \) decide whether \( F \) is satisfiable by all truth assignments.

We also consider the following problems on boolean circuits. Recall that two circuits \( C_1(x_1, \ldots, x_n) \) and \( C_2(x_1, \ldots, x_n) \) are isomorphic if there is a permutation \( \pi \) of \( \{1, \ldots, n\} \) such that, for any truth assignment \( x \in \{0,1\}^n \), \( C_1(x) = C_2(\pi(x)) \).
Boolean circuit isomorphism problem (CircuitIso): Given two boolean circuits $C_1$ and $C_2$ decide whether $C_1$ and $C_2$ are isomorphic.

A related problem is based on the notion of congruence. A congruence between two circuits on $n$ variables, $C_1(x_1, \ldots, x_n)$ and $C_2(x_1, \ldots, x_n)$ is a mapping $\psi = (\pi, f_1, \ldots, f_n)$, where $\pi$ is a permutation of $\{1, \ldots, n\}$ and, for any $1 \leq i \leq n$, $f_i$ is a permutation on $\{0, 1\}$ (either the identity or the negation function). As in the case of game morphism, the image $\psi(x)$ is obtained by permuting the positions of the input bits, according to permutation $\pi$, and then applying to any bit $i$ the permutation $f_i$.

Boolean circuit congruence problem (CircuitCong): Given two circuits $C_1$ and $C_2$ decide whether $C_1$ and $C_2$ are congruent.

The CircuitIso problem has been studied by B. Borchert, D. Ranjan and F. Stephan in [3], among many other results they show that CircuitIso and CircuitCong are equivalent. It is known that CircuitIso $\in \Sigma^p_2$, but it cannot be $\Sigma^p_2$-hard unless the polynomial hierarchy collapses [1].

Two graphs are isomorphic if there is a one-to-one correspondence between their vertices and there is an edge between two vertices of one graph if and only if there is an edge between the two corresponding vertices in the other graph.

Graph isomorphism (GI): Given two graphs, decide whether they are isomorphic.

It is well known that GI is not expected to be NP-hard [9]. Let us start with the complexity for IsIso problem in the case of strategic games.

Theorem 1. The IsIso and the IsAuto problems for strong morphisms are coNP-complete when the games are given in general form. Both problems belong to NC whenever the games are given in explicit form. The strong isomorphism is given in both cases in explicit form.

Our next step is to provide upper bounds for the complexity of the Iso and the Auto problems. Later on we show that the bounds are best possible for the Iso problems.

Theorem 2. The Iso and the Auto problems for strong morphisms belong to $\Sigma^p_2$ when the games are given in general form. Both problems belong to NP when the games are given in explicit form.

We prove that Iso is equivalent to CircuitIso for games in general form. This is done through a series of reductions transforming the game while preserving the existence of strong isomorphism. First, we show how to construct a game in which the set of actions for each player is $\{0, 1\}$, which we call a binary action game. Second, we show how to construct from a binary action game another binary action game in which the utility functions range is $\{0, 1\}$, which we call a binary game. Finally, we show the equivalence with the Boolean circuit congruence. All the transformations presented in the paper can be computed in polynomial time, thus we avoid to mention this fact all through the paper. Let us start with the first transformation.
Let \( I = (N, (A_i)_{i \in N}, (u_i)_{i \in N}) \) be a game and let \( \mu_I = \min\{u_i(a) | a \in A, i \in N\} \) be the smaller payoff obtained by any player, take \( \mu < \mu_I \) and call such value \( \mu \) a “penalty payoff”. Without loss of generality we assume that \( N = \{1, \ldots, n\} \) and that, for any \( i \in N, A_i = \{1, \ldots, k_i\} \) for suitable values. Given \( A_i = \{1, \ldots, k_i\} \) we “binify” an action \( j \in A_i \) coding it with \( k_i \) bits, defined by \( \text{binify}(j) = 0^{j-1}10^{k_i-j} \). Moreover we associate with \( A_i \) a block \( B_i \) of \( k_i \) players each one taking care of one bit. In this case we get \( k = \sum_{i \in N} k_i \) players partitioned into \( B_1, \ldots, B_n \) blocks. Given \( i \in B_j \) we say that \( i \) belongs to block \( j \) of players and write \( \text{block}(i) = j \). The binify process can be used in a strategy profile to clarify notation, given \( a = (a_1, \ldots, a_n) \), we write \( \text{binify}(a) = (\text{binify}(a_1), \ldots, \text{binify}(a_n)) \). Often we look at \( \text{binify}(a) \) as a \( k \) tuple of bits. For instance, given \( I \) with 3 players \( A_1 = A_2 = \{1, 2\} \) and \( A_2 = \{1, 2, 3\} \) we have \( \text{binify}(1,2,2) = (10,010,01) = (1,0,1,0,0,1) \). Set \( A' = \{0,1\}^k \), as for any \( a \in A \) it holds \( \text{binify}(a) \in A' \), we can define \( \text{good}(A') = \{ \text{binify}(a) | a \in A \} \) and \( \text{bad}(A') = A' \setminus \text{good}(A) \). When \( a' \in \text{good}(A) \) we say that \( a' \) is good, otherwise is bad. Note that \( \text{binify} : A \rightarrow \text{good}(A') \) is a bijection, therefore the inverse function is also a bijection, for instance \( \text{binify}^{-1}(1,0,1,0,0,1) = (1,2,2) \).

**BinaryAct**\((I, \mu)\) is defined as

\[
\text{BinaryAct}(I, \mu) = (N', (A'_i)_{i \in N'}, (u_i')_{i \in N'})
\]

where \( N' = \{1, \ldots, k\} \) and, for any \( i \in N', A'_i = \{0,1\} \) and thus the set of action profiles is \( A' = \{0,1\}^k \). The utilities are defined by

\[
u'_i(a') = \begin{cases} u_{\text{block}(i)}(\text{binify}^{-1}(a')) & \text{if } a' \in \text{good}(A'), \\ \mu & \text{if } a' \in \text{bad}(A'). \end{cases}
\]

Notice that, for \( a \in A \), \( u'_i(\text{binify}(a)) = u_{\text{block}(i)}(a) \), furthermore, all the players in a given block have the same utility. Each strategy profile \( a' \) in **BinaryAct**\((I, \mu)\) can be factorized giving the actions taken by the \( k \) players as \( a' = (a'_1, \ldots, a'_k) \) or grouping the actions according to teams \( B_1, \ldots, B_n \) as \( a' = (b_1, \ldots, b_n) \) where \( b_i \) is a strategy profile for \( B_i \). The big gap in the utility function is used to created a gap that separates the profiles in **BinaryAct**\((I, \mu)\) that codify correctly a profile of \( I \) from those that do not.

**Lemma 1.** Let \( I_1, I_2 \) be two games given in general form and set \( t = \max\{t_1, t_2\}, \)

\( t_i \) for \( 1 \leq i \leq 2 \), is the time allowed to the utility TM of the game \( I \). There is a strong isomorphism between \( I_1 \) and \( I_2 \) iff there is a strong isomorphism between the games **BinaryAct**\((I_1, \mu)\) and **BinaryAct**\((I_2, \mu)\) where \( \mu = -2^t \).

Let us now transform a binary actions game into a binary game. Given a game \( I = (N, (A_i)_{i \in N}, (u_i)_{i \in N}) \) in which \( A_i = \{0,1\} \), for any \( i \in N, \) \( N = \{1, \ldots, n\} \). Given positive values \( t \) and \( m \) such that, for any action profile \( a \) and any player \( i, |u_i(a)| \leq t \) and \( m \geq \{n, t\} \). We set \( k = n + tn + m + 2 \).

**Binary**\((I, t, m)\) is defined as

\[
\text{Binary}(I, t, m) = (N', (A'_i)_{i \in N'}, (u'_i)_{i \in N'})
\]

where \( N' = \{1, \ldots, k\} \) and, for any \( i \in N' \), \( A'_i = \{0, 1\} \).

Before defining the utilities we need some additional notation. The set \( N' \) is partitioned into \( n + 2 \) consecutive intervals \( B_0, \ldots, B_n, B_{n+1} \) so that the interval \( B_0 \) has exactly \( n \) players, for \( 1 \leq i \leq n \), the block \( B_i \) has \( t \) players, finally block \( B_{n+1} \) has \( m + 2 \) players. Inside the blocks we use relative coordinates to identify the players. In all the blocks coordinates start at 1 except for the last block that starts with 0. In this situation a strategy profile \( a \) is usually factorized as \( a = x b_1 \ldots b_n z \) where \( x = x_1 \ldots x_n \), \( b_i = b_{i1} \ldots b_{it} \), and \( z = z_0 \ldots z_{m+1} \). We define the utility function by properties of the strategy profile, assume that \( a = x b_1 \ldots b_n z \) is a strategy profile of \( \text{BINARY}(\Gamma, t, m) \).

- In the case that, for some \( \ell \), \( 0 \leq \ell \leq m + 1 \), the last \( \ell \) bits of \( a \) are 1, all the players except the last \( \ell \) get utility 0. The remaining players get utility 1.
- In the case that, for some \( j \), \( 1 \leq j \leq t \), the \( j \)-th bit of \( a \) is the unique 1 in \( z \), all the players in blocks \( B_1, \ldots, B_n \) that do not occupy position \( j \) in their block get utility 0, all the players in blocks \( B_0 \) and \( B_{n+1} \) get utility 1, all the remaining players get as utility their action.
- In the case that, the 0-th bit of \( a \) is the unique 1 in \( z \), for any \( i \), \( 1 \leq i \leq n \), player \( i \) and all the players in block \( B_i \) get utility 1 when \( u_i(x) = b^i \). All the players in block \( B_{n+1} \) get utility 0.
- In the remaining cases all the players get utility 1.

Notice that the utilities for all the players are either 0 or 1.

**Lemma 2.** Let \( \Gamma_1, \Gamma_2 \) be two games given in general form, set \( t = \max\{t_1, t_2, 3\} \), where \( t_i \) is the time allowed to the utility \( \Gamma_i \) of game \( \Gamma_i \), and \( m = \max\{t, n_1, n_2\} \), where \( n_i \) is the number of players in game \( \Gamma_i \). There is a strong isomorphism between \( \Gamma_1 \) and \( \Gamma_2 \) iff there is a strong isomorphism between \( \text{BINARY}(\Gamma_1, t, m) \) and \( \text{BINARY}(\Gamma_2, t, m) \).

Given a binary game \( \Gamma = (N, (A_i))_{i \in N}, (u_i))_{i \in N} \) with \( n \) players, such that for any \( 1 \leq i \leq n \), utility \( u_i \) has range \( \{0, 1\} \) and \( A_i = \{0, 1\} \). We construct a circuit \( C_\Gamma \) on \( 4n + 2 \) variables. Remind that, when \( u_i(x) \) is computed by a Turing machine in polynomial time, Ladner’s construction [15] gives us a polynomial size circuit.

**Circuit \( C_\Gamma \).** The variables in \( C_\Gamma \) are grouped in four blocks, the \( X \)-block contains the first \( n \)-variables, the \( Y \)-block is formed by the variables in positions \( n+1 \) to \( 2n \), the \( C \)-block contain the following \( n + 2 \) variables, and the \( D \)-block the remaining variables. For sake of readability we split the set of variables into four parts \( a = (x, y, c, d) \) where \( x = (x_1, \ldots, x_n) \), \( y = (y_1, \ldots, y_n) \), \( c = (c_1, \ldots, c_{n+2}) \), and \( d = (d_1, \ldots, d_n) \).

We define \( C_\Gamma \) with the help of \( n + 2 \) following circuits.

\[
C_1(x, y, d) = [(x_1 = \overline{d_1}) \land \cdots \land (x_n = \overline{d_n}) \land (u_1(x) = y_1) \land \cdots \land (u_n(x) = y_n)]
\]

\[
C_2(y) = [y_1 \lor \cdots \lor y_n]
\]

\[
C_{i+2}(x_i, y_i, d_i) = \lceil \overline{x_i} \land (x_i = \overline{d_i}) \rceil \quad \text{for} \quad 1 \leq i \leq n.
\]

Finally

\[
C_\Gamma(x, y, c, d) = \begin{cases} 0 & \text{if } \sum_{1 \leq i \leq n+2} c_i = 0 \text{ or } \sum_{1 \leq i \leq n+2} c_i > 1 \\ C_j & \text{if } \sum_{1 \leq i \leq n+2} c_i = 1 \text{ and } c_j = 1 \end{cases}
\]
The previous construction is used to reduce the ISO problem to the CIRCUIT-CONG problem.

Lemma 3. Let $\Gamma$ and $\Gamma'$ be two games in general form with at least two players each, boolean utilities and actions $\{0, 1\}$. There is a congruence isomorphism between $C_\Gamma$ and $C_{\Gamma'}$ iff there is a strong isomorphism between $\Gamma$ and $\Gamma'$.

Proving NP-completeness in the case of explicit form appears to be a difficult task. Observe, that a game in explicit form can be seen as a graph with edge labels and weights. As the total number of different weights appearing in both games is polynomial the problem can be reduced to the Graph isomorphism (GI) problem [17]. Therefore the NP-hardness of ISO will imply the NP-hardness of GI. Our proof provides a reduction that shows that the opposite direction is true. It is easy to show that CIRCUIT-CONG is reducible to ISO, just consider a game with as many players as variables in which the utilities for all the players are identical and coincide with the evaluation of the circuit. Taking into account that CIRCUIT-CONG is equivalent to CIRCUIT-Iso and putting all together we have:

Theorem 3. The ISO problem for strong isomorphism and games given in general form is equivalent to the circuit isomorphism problem. In the case of games given in extensive form the problem is equivalent to graph isomorphism.

4 Weak and Local Isomorphism

There are several ways to relax the notion of strong isomorphism while maintaining the structure of Nash equilibria. For instance, Harsanyi and Selten [8] substitute $u_{\pi(i)}(\psi(a)) = u_i(a)$ for $u_{\pi(i)}(\psi(a)) = \alpha_i u_i(a) + \beta_i$. In order to generalize this approach we consider, following [12], games in which utility functions are replaced by preference relations ($\preceq_i \in N$). All the preference relations must be total, that is, given any pair $a, a'$ holds $a \preceq_i a'$ or $a' \preceq_i a$. In this case, a game is a tuple $\Gamma = (N, (A_i)_{i \in N}, (\preceq_i)_{i \in N})$. We note strict preference as usual, $a \prec_i a'$ iff $a \preceq_i a'$ but not $a' \preceq_i a$. We note indifference by $a' \sim_i a'$, as usual indifference occurs when $a \preceq_i a'$ and $a' \preceq_i a$ holds. The definition of isomorphism can be adapted to respect preference relations instead of utility functions.

Definition 4. A weak isomorphism $\psi : \Gamma \rightarrow \Gamma'$ is a mapping $\psi = (\pi, (\varphi_i)_{i \in N})$ such that any triple $a, a'$ and $i$ verifies:

\[
\begin{align*}
\text{Preserve}(a, a', \psi, i) & \equiv \\
(a \prec_i a' \Rightarrow \psi(a) \prec_{\pi(i)} \psi(a')) & \lor (a \sim_i a' \Rightarrow \psi(a) \sim_{\pi(i)} \psi(a')).
\end{align*}
\]

Preference relations can be defined using utility functions, $a \prec_i a'$ iff $u_i(a) < u_i(a')$ and $a \sim_i a'$ iff $u_i(a) = u_i(a')$.

Weak isomorphisms preserves preferences for any pair of strategy profiles and therefore maintains the structure of pure Nash equilibria. However, if we are
interested to maintain this structure through isomorphisms we need to consider only preferences \( a \preceq_i a' \) such that \( a - i = a' - i \). We call these preferences local preferences.

**Definition 5.** A local isomorphism \( \psi : \Gamma \to \Gamma' \) is a mapping \( \psi \) such that for any triple \( a, a', i \) such that \( a - i = a' - i \) verifies \( \text{Preserve}(a, a', \psi, i) \).

It is easy to see that weak and local isomorphisms preserve pure Nash equilibria.

In order to describe games in general form with total or local preferences we consider the following

\[
M_\preceq(i, a, a') = \begin{cases} 
    i & \text{if } a \sim_i (a_{-i}, a'_i) \text{ (indifference case)} \\
    b & \text{if } a \prec_i (a_{-i}, a'_i) \text{ (better case)} \\
    w & \text{if } a \succ_i (a_{-i}, a'_i) \text{ (worse case)}
\end{cases}
\]

Given \( \Gamma = \langle 1^n, A_1, \ldots, A_n, M, 1^t \rangle \) such that \( M(i, a) = u_i(a) \) we can easily build \( \Gamma' = \langle 1^n, A_1, \ldots, A_n, M_\preceq, 1^t \rangle \) in polynomial time. In order to describe games in explicit form we consider a tuple \( \Gamma = \langle 1^n, A_1, \ldots, A_n, M_\preceq, L_\succ, L_\sim \rangle \) where \( L_\succ \) and \( L_\sim \) are two adjacency lists. The list \( L_\succ[a][i] = (a'_{i_1}, a'_{i_2}, \ldots, a'_{i_\ell}) \) stores all the actions \( a_{i_j} \in A_i \) such that \( a \succ_i (a_{-i}, a'_{i_j}) \). and \( L_\sim \) store the elements \( a_{i_j} \) such that \( a \sim_i (a_{-i}, a'_{i_j}) \). Replacing strict by weak or local isomorphisms does not modify complexity bounds.

**Theorem 4.** In the case of weak or local isomorphisms, the \( \text{IsIso} \) is \( \text{coNP}\)-complete, for games given in general form, and it belongs to \( \text{NC} \) when the games are given in explicit form. The \( \text{Iso} \) problem belongs to \( \Sigma^p_2 \), when the games are given in general form and it belongs to \( \text{NP} \) when the games are given in explicit form.

In the case of \( \text{Iso} \) we can also prove

**Theorem 5.** The \( \text{Iso} \) problem for weak and local isomorphism and games given in general form is equivalent to the circuit isomorphism problem. In the case of games given in explicit form the problem is equivalent to graph isomorphism.

## 5 Small Case Study

An interesting problem is to have a classification of strategic games with the same number of players according to the structure of the pure Nash equilibria. This requires the development of acceptable definitions of equivalence between games. A first naive approach is to consider games as equivalent if they have the same number of Nash equilibria. This approach has been undertaken via probabilistic analysis by I. Y. Powers [14]. She studied the limit distributions of the number of pure strategies Nash equilibria for \( n \) players strategic games. Further results in [16].
Observe that for 2-players each one with $m$ actions we get a bimatrix with $m^2$ positions. As numbers $\{0, \ldots, m^2 - 1\}$ are enough to encode preferences, there are $m^{4m^2}$ different bimatrix games. In the particular case of 2 players and 2 actions the games can be grouped, according to the number of Nash equilibria, as follows:

<table>
<thead>
<tr>
<th>Number of PNE</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Games</td>
<td>2592</td>
<td>29376</td>
<td>27936</td>
<td>5376</td>
<td>256</td>
<td>65536</td>
</tr>
</tbody>
</table>

The above table has been obtained by exhaustive computation. For bigger $m$ this analysis becomes quickly intractable.

This simple approach has limitations because games with different structure can have the same number of Nash equilibria. Therefore there is a need to provide a better answer to the question: When are two games the same? The obvious mathematical setting to deal with equivalence is through isomorphisms. When games are defined with preferences, we consider weak isomorphisms. Two games are equivalent iff they are weakly isomorphic and we partition the set of 65536 games into equivalence classes.

In the case of games with two players and two actions, the set of strategy profiles is $\{00, 01, 10, 11\}$. This set can be represented as the set of nodes of a square where edges represent the preferences. We note, for instance $00 \sim 01$ to mean $00 \sim_2 01$ and $00 \rightarrow 01$ iff $00 \prec_2 01$ (when dealing with local preferences, the subindex 2 can be avoided). Therefore, each class of equivalence is represented by a square fulfilling some conditions about preferences. In figure 2 we provide all the equivalence classes, giving information about the possible structure of the Nash equilibria. In fact, as one can see, the number is much smaller than 65536.

Therefore we have more concise information about the structure of the Nash equilibria. The graphs appearing in Figure 2 are close to the Nash dynamics graphs defined in [5], observe that here we are considering double connections between equally likely profiles.

Fig. 2. Different game classes.
6 Comments and Open Questions

The equivalence between the ISO problem for games in general form and boolean circuit equivalence, has been obtained using the simulation of Turing machines by circuits. For families of games whose utility functions are defined by boolean formulas, the ISO problem turns out to be equivalent to boolean formula equivalence. Recall that boolean formula equivalence is not believed to be $\Sigma_p^2$-hard [1], and that it is still open whether the formula isomorphism is equivalent to circuit isomorphism.

We are working towards extending the definitions of game isomorphism to extensive games avoiding the use of strategic forms. An interesting open question is defining game isomorphisms for games without perfect information.

References