On amending the sufficient conditions for Nash implementation

Wu, Haoyang

9 August 2011

On amending the sufficient conditions for Nash implementation

Haoyang Wu *

Abstract

The Maskin's theorem is a fundamental work in the theory of mechanism design. A recent work [Wu, Quantum mechanism helps agents combat "bad" social choice rules. *Inter. J. of Quantum Information* 9 (2011) 615-623] shows that when an additional condition is satisfied, the Maskin's theorem will no longer hold if agents use quantum strategies. Although quantum mechanisms are theoretically feasible, they are not applicable to the macro world immediately due to the restriction of current experimental technologies. In this paper, we will go beyond the obstacle of how to realize quantum mechanisms, and propose an algorithmic mechanism which amends the sufficient conditions of the Maskin's theorem immediately just in the macro world (i.e., computer or Internet world).

Key words: Algorithmic mechanism; Mechanism design; Nash implementation.

1 Introduction

Nash implementation is the cornerstone of the theory of mechanism design. The Maskin's theorem describes the sufficient conditions for Nash implementation (i.e., monotonicity and no-veto power) when the number of agents are at least three [1]. Since a social choice rule (SCR) is specified by a designer, a desired outcome for the designer may not be the most favorite one for the agents (See Example 1 of Ref. [2]).

According to the Maskin's theorem, given an SCR that is monotonic and satisfies no-veto, it is impossible for the agents to fight the designer even if all agents dislike the SCR. However, in 2011, Wu [2] generalized the theory

* Wan-Dou-Miao Research Lab, Suite 1002, 790 WuYi Road, Shanghai, 200051, China.

Email address: `hywch@mail.xjtu.edu.cn`, `Tel: 86-18621753457` (Haoyang Wu).

of mechanism design to a quantum domain and proposed that by virtue of quantum strategies, agents who satisfied a certain condition could combat Pareto-inefficient SCRs instead of being restricted by the Maskin's theorem. For n agents, the time and space complexity of the quantum mechanism are $O(n)$. Therefore the quantum mechanism is theoretically feasible.

It should be emphasized that the so-called quantum mechanism does not mean that the agents have the designer to design a new mechanism to attain Pareto superior outcomes. Indeed, from the viewpoint of the designer, nothing has been changed. As a comparison, from the viewpoint of the agents, the manner in which they participate the mechanism will be changed when they use quantum strategies. In other words, in the quantum mechanism, the agents participate the traditional Maskin's mechanism with a richer strategy space.

Despite these interesting results, there exists an obstacle for agents to use the quantum mechanism in the macro world immediately: It needs a quantum equipment to work, but so far the experimental technologies for quantum information are not commercially available [3]. As a result, the quantum mechanism may be viewed only as a "toy". In this paper, we will go beyond this obstacle and propose an algorithmic mechanism which amends the sufficient conditions for Nash implementation just in the macro world (The main result is Proposition 1 in Section 3.4). The rest of the paper is organized as follows: Section 2 recalls preliminaries of classical and quantum mechanisms published in Refs. [4,2] respectively; Section 3 is the main part of this paper. Section 4 draws conclusions.

2 Preliminaries

2.1 The classical theory of mechanism design [4]

Let $N = \{1, \cdots, n\}$ be a finite set of *agents* with $n \geq 2$, $A = \{a_1, \cdots, a_k\}$ be a finite set of social *outcomes*. Let T_i be the finite set of agent i's types, and the *private information* possessed by agent i is denoted as $t_i \in T_i$. We refer to a profile of types $t = (t_1, \cdots, t_n)$ as a *state*. Let $\mathcal{T} = \prod_{i\in N} T_i$ be the set of states. At state $t \in \mathcal{T}$, each agent $i \in N$ is assumed to have a complete and transitive *preference relation* $\succeq_i^t$ over the set A. We denote by $\succeq^t = (\succeq_1^t, \cdots, \succeq_n^t)$ the profile of preferences in state t, and denote by $\succ_i^t$ the strict preference part of $\succeq_i^t$. Fix a state t, we refer to the collection $E =< N, A, (\succeq_i^t)_{i\in N} >$ as an *environment*. Let ε be the class of possible environments. A *social choice rule* (SCR) F is a mapping $F : \varepsilon \to 2^A \backslash \{\emptyset\}$. A *mechanism* $\Gamma = ((M_i)_{i\in N}, g)$ describes a message or strategy set M_i for agent i, and an outcome function $g : \prod_{i\in N} M_i \to A$. M_i is unlimited except that if a mechanism is direct,

$M_i = T_i$.

An SCR F satisfies *no-veto* if, whenever $a \succeq_i^t b$ for all $b \in A$ and for all agents i but perhaps one j, then $a \in F(E)$. An SCR F is *monotonic* if for every pair of environments E and E', and for every $a \in F(E)$, whenever $a \succeq_i^t b$ implies that $a \succeq_i^{t'} b$, there holds $a \in F(E')$. We assume that there is *complete information* among the agents, i.e., the true state t is common knowledge among them. Given a mechanism $\Gamma = ((M_i)_{i \in N}, g)$ played in state t, a *Nash equilibrium* of Γ in state t is a strategy profile m^* such that: $\forall i \in N, g(m^*(t)) \succeq_i^t g(m_i, m_{-i}^*(t)), \forall m_i \in M_i$. Let $\mathcal{N}(\Gamma, t)$ denote the set of Nash equilibria of the game induced by Γ in state t, and $g(\mathcal{N}(\Gamma, t))$ denote the corresponding set of Nash equilibrium outcomes. An SCR F is *Nash implementable* if there exists a mechanism $\Gamma = ((M_i)_{i \in N}, g)$ such that for every $t \in \mathcal{T}$, $g(\mathcal{N}(\Gamma, t)) = F(t)$.

Maskin [1] provided an almost complete characterization of SCRs that were Nash implementable. The main results of Ref. [1] are two theorems: 1) (*Necessity*) If an SCR is Nash implementable, then it is monotonic. 2) (*Sufficiency*) Let $n \geq 3$, if an SCR is monotonic and satisfies no-veto, then it is Nash implementable. In order to facilitate the following investigation, we briefly recall the Maskin's mechanism published in Ref. [4] as follows:

Consider the following mechanism $\Gamma = ((M_i)_{i \in N}, g)$, where agent i's message set is $M_i = A \times \mathcal{T} \times \mathbb{Z}_+$, where $\mathbb{Z}_+$ is the set of non-negative integers. A typical message sent by agent i is described as $m_i = (a_i, t_i, z_i)$. The outcome function g is defined in the following three rules: (1) If for every agent $i \in N$, $m_i = (a, t, 0)$ and $a \in F(t)$, then $g(m) = a$. (2) If $(n-1)$ agents $i \neq j$ send $m_i = (a, t, 0)$ and $a \in F(t)$, but agent j sends $m_j = (a_j, t_j, z_j) \neq (a, t, 0)$, then $g(m) = a$ if $a_j \succ_j^t a$, and $g(m) = a_j$ otherwise. (3) In all other cases, $g(m) = a'$, where a' is the outcome chosen by the agent with the lowest index among those who announce the highest integer.

2.2 Quantum mechanisms [2]

In 2011, Wu [2] combined the theory of mechanism design with quantum mechanics and found that when an additional condition was satisfied, monotonicity and no-veto are not sufficient conditions for Nash implementation in the context of a quantum domain. Following Section 4 in Ref. [2], two-parameter quantum strategies are drawn from the set:

$$\hat{\omega}(\theta, \phi) \equiv \begin{bmatrix} e^{i\phi}\cos(\theta/2) & i\sin(\theta/2) \\ i\sin(\theta/2) & e^{-i\phi}\cos(\theta/2) \end{bmatrix}, \tag{1}$$

$\hat{\Omega} \equiv \{\hat{\omega}(\theta,\phi) : \theta \in [0,\pi], \phi \in [0,\pi/2]\}$, $\hat{J} \equiv \cos(\gamma/2)\hat{I}^{\otimes n} + i\sin(\gamma/2)\hat{\sigma_x}^{\otimes n}$ (where $\gamma \in [0,\pi/2]$ is an entanglement measure), $\hat{I} \equiv \hat{\omega}(0,0)$, $\hat{D}_n \equiv \hat{\omega}(\pi,\pi/n)$, $\hat{C}_n \equiv \hat{\omega}(0,\pi/n)$.

According to the last paragraph of page 392 in Ref. [4], here we also assume there is complete information among agents. Put differently, there is no private information for any agent. Without loss of generality, we assume that:
1) Each agent i has a quantum coin i (qubit) and a classical card i. The basis vectors $|C\rangle = (1,0)^T$, $|D\rangle = (0,1)^T$ of a quantum coin denote head up and tail up respectively.
2) Each agent i independently performs a local unitary operation on his/her own quantum coin. The set of agent i's operation is $\hat{\Omega}_i = \hat{\Omega}$. A strategic operation chosen by agent i is denoted as $\hat{\omega}_i \in \hat{\Omega}_i$. If $\hat{\omega}_i = \hat{I}$, then $\hat{\omega}_i(|C\rangle) = |C\rangle$, $\hat{\omega}_i(|D\rangle) = |D\rangle$; If $\hat{\omega}_i = \hat{D}_n$, then $\hat{\omega}_i(|C\rangle) = |D\rangle$, $\hat{\omega}_i(|D\rangle) = |C\rangle$. $\hat{I}$ denotes "*Not flip*", $\hat{D}_n$ denotes "*Flip*".
3) The two sides of a card are denoted as Side 0 and Side 1. The message written on the Side 0 (or Side 1) of card i is denoted as $card(i,0)$ (or $card(i,1)$). A typical card written by agent i is described as $c_i = (card(i,0), card(i,1))$. The set of c_i is denoted as C_i.
4) There is a device that can measure the state of n coins and send messages to the designer.

A *quantum mechanism* $\Gamma^Q = ((\hat{S}_i)_{i\in N}, \hat{G})$ describes a strategy set $\hat{S}_i = \hat{\Omega}_i \times C_i$ for each agent i and an outcome function $\hat{G} : \otimes_{i\in N}\hat{\Omega}_i \times \prod_{i\in N} C_i \to A$. We use $\hat{S}_{-i}$ to express $\otimes_{j\neq i}\hat{\Omega}_j \times \prod_{j\neq i} C_j$, and thus, a strategy profile is $\hat{s} = (\hat{s}_i, \hat{s}_{-i})$, where $\hat{s}_i \in \hat{S}_i$ and $\hat{s}_{-i} \in \hat{S}_{-i}$. The strategic behavior of each agent i is to strategically choose $\hat{\omega}_i$, $card(i,0)$ and $card(i,1)$.

A *Nash equilibrium* of a quantum mechanism Γ^Q played in state t is a strategy profile $\hat{s}^* = (\hat{s}_1^*, \cdots, \hat{s}_n^*)$ such that for any agent $i \in N$ and $\hat{s}_i \in \hat{S}_i$, $\hat{G}(\hat{s}_1^*, \cdots, \hat{s}_n^*) \succeq_i^t \hat{G}(\hat{s}_i, \hat{s}_{-i}^*)$. The setup of a quantum mechanism $\Gamma^Q = ((\hat{S}_i)_{i\in N}, \hat{G})$ is depicted in Fig. 1. The working steps of the quantum mechanism Γ^Q are given as follows (with slight differences from Ref. [2]):

Step 1: The state of every quantum coin is set as $|C\rangle$. The initial state of the n quantum coins is $|\psi_0\rangle = \underbrace{|C\cdots CC\rangle}_{n}$.

Step 2: Given a state t, if two following conditions are satisfied, goto Step 4:
1) There exists $\hat{t} \in \mathcal{T}$, $\hat{t} \neq t$ such that $\hat{a} \succeq_i^t a$ (where $\hat{a} \in F(\hat{t})$, $a \in F(t)$) for every $i \in N$, and $\hat{a} \succ_j^t a$ for at least one $j \in N$;
2) If there exists $\hat{t}' \in \mathcal{T}$, $\hat{t}' \neq \hat{t}$ that satisfies the former condition, then $\hat{a} \succeq_i^t \hat{a}'$ (where $\hat{a} \in F(\hat{t})$, $\hat{a}' \in F(\hat{t}')$) for every $i \in N$, and $\hat{a} \succ_j^t \hat{a}'$ for at least one $j \in N$.
Step 3: Each agent i sets $c_i = ((a_i, t_i, z_i), (a_i, t_i, z_i))$ (where $a_i \in A$, $t_i \in \mathcal{T}$, $z_i \in \mathbb{Z}_+$), $\hat{\omega}_i = \hat{I}$. Goto Step 7.

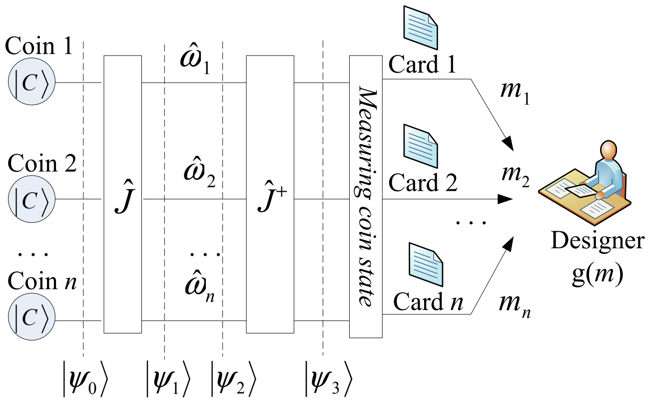

Fig. 1. The setup of a quantum mechanism. Each agent has a quantum coin and a card. Each agent independently performs a local unitary operation on his/her own quantum coin.

Step 4: Each agent i sets $c_i = ((\hat{a}, \hat{t}, 0), (a_i, t_i, z_i))$. Let n quantum coins be entangled by $\hat{J}$. $|\psi_1\rangle = \hat{J}|C \cdots CC\rangle$.
Step 5: Each agent i independently performs a local unitary operation $\hat{\omega}_i$ on his/her own quantum coin. $|\psi_2\rangle = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}|C \cdots CC\rangle$.
Step 6: Let n quantum coins be disentangled by $\hat{J}^{+}$. $|\psi_3\rangle = \hat{J}^{+}[\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}|C \cdots CC\rangle$.
Step 7: The device measures the state of n quantum coins and sends $card(i, 0)$ (or $card(i, 1)$) as a message m_i to the designer if the state of quantum coin i is $|C\rangle$ (or $|D\rangle$).
Step 8: The designer receives the overall message $m = (m_1, \cdots, m_n)$ and let the final outcome be $g(m)$ using rules (1)-(3) of the Maskin's mechanism. END.

3 Main results

3.1 Matrix representations of quantum states

In quantum mechanics, a quantum state can be described as a vector. For a two-level system, there are two basis vectors: $(1, 0)^T$ and $(0, 1)^T$. The matrix representations of quantum states $|\psi_0\rangle$, $|\psi_1\rangle$, $|\psi_2\rangle$ and $|\psi_3\rangle$ are given as follows.

$$|C\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \hat{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \hat{\sigma}_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, |\psi_0\rangle = \underbrace{|C \cdots CC\rangle}_{n} = \begin{bmatrix} 1 \\ 0 \\ \cdots \\ 0 \end{bmatrix}_{2^n \times 1} \tag{2}$$

$$\hat{J} = \cos(\gamma/2)\hat{I}^{\otimes n} + i\sin(\gamma/2)\hat{\sigma}_x^{\otimes n} \tag{3}$$

$$= \begin{bmatrix} \cos(\gamma/2) & & & & & i\sin(\gamma/2) \\ & \cdots & & & \cdots & \\ & & \cos(\gamma/2) & i\sin(\gamma/2) & & \\ & & i\sin(\gamma/2) & \cos(\gamma/2) & & \\ & \cdots & & & \cdots & \\ i\sin(\gamma/2) & & & & & \cos(\gamma/2) \end{bmatrix}_{2^n \times 2^n} \tag{4}$$

For $\gamma = \pi/2$,

$$\hat{J}_{\pi/2} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & & & & & i \\ & \cdots & & & \cdots & \\ & & 1 & i & & \\ & & i & 1 & & \\ & \cdots & & & \cdots & \\ i & & & & & 1 \end{bmatrix}_{2^n \times 2^n} \tag{5}$$

$$|\psi_1\rangle = \hat{J}\underbrace{|C \cdots CC\rangle}_{n} = \begin{bmatrix} \cos(\gamma/2) \\ 0 \\ \cdots \\ 0 \\ i\sin(\gamma/2) \end{bmatrix}_{2^n \times 1} \tag{6}$$

Following formula (1), we define:

$$\hat{\omega}_1 = \begin{bmatrix} e^{i\phi_1}\cos(\theta_1/2) & i\sin(\theta_1/2) \\ i\sin(\theta_1/2) & e^{-i\phi_1}\cos(\theta_1/2) \end{bmatrix}, \cdots, \hat{\omega}_n = \begin{bmatrix} e^{i\phi_n}\cos(\theta_n/2) & i\sin(\theta_n/2) \\ i\sin(\theta_n/2) & e^{-i\phi_n}\cos(\theta_n/2) \end{bmatrix}, \tag{7}$$

The dimension of $\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n$ is $2^n \times 2^n$. Since only two values in $|\psi_1\rangle$ are non-zero, it is not necessary to calculate the whole $2^n \times 2^n$ matrix to obtain $|\psi_2\rangle$. Indeed, we only need to calculate the leftmost and rightmost column of $\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n$ to derive $|\psi_2\rangle = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}\underbrace{|C\cdots CC\rangle}_{n}$.

$$\hat{J}^+ = \begin{bmatrix} \cos(\gamma/2) & & & & & -i\sin(\gamma/2) \\ & \cdots & & & \cdots & \\ & & \cos(\gamma/2) & -i\sin(\gamma/2) & & \\ & & -i\sin(\gamma/2) & \cos(\gamma/2) & & \\ & \cdots & & & \cdots & \\ -i\sin(\gamma/2) & & & & & \cos(\gamma/2) \end{bmatrix}_{2^n \times 2^n} \tag{8}$$

$$|\psi_3\rangle = \hat{J}^+|\psi_2\rangle \tag{9}$$

3.2 A simulating algorithm

Based on the aforementioned matrix representations of quantum states, in the following we will propose a simulating algorithm that simulates the quantum operations and measurements in Steps 4-7 of the quantum mechanism given in Section 2.2. Since the entanglement measurement γ is just a control factor, γ can be simply set as its maximum $\pi/2$. For n agents, the inputs and outputs of the simulating algorithm are illustrated in Fig. 2. The *Matlab* program is given in Fig. 3(a)-(d).

Inputs:
1) θ_i, ϕ_i, $i = 1, \cdots, n$: the parameters of agent i's local operation $\hat{\omega}_i$, $\theta_i \in [0, \pi]$, $\phi_i \in [0, \pi/2]$.
2) $card(i, 0), card(i, 1)$, $i = 1, \cdots, n$: the information written on the two sides of agent i's card, where $card(i, 0) = (a_i, t_i, z_i) \in A \times \mathcal{T} \times \mathbb{Z}_+$, $card(i, 1) = (a'_i, t'_i, z'_i) \in A \times \mathcal{T} \times \mathbb{Z}_+$.

Outputs:
m_i, $i = 1, \cdots, n$: the agent i's message that is sent to the designer, $m_i \in$

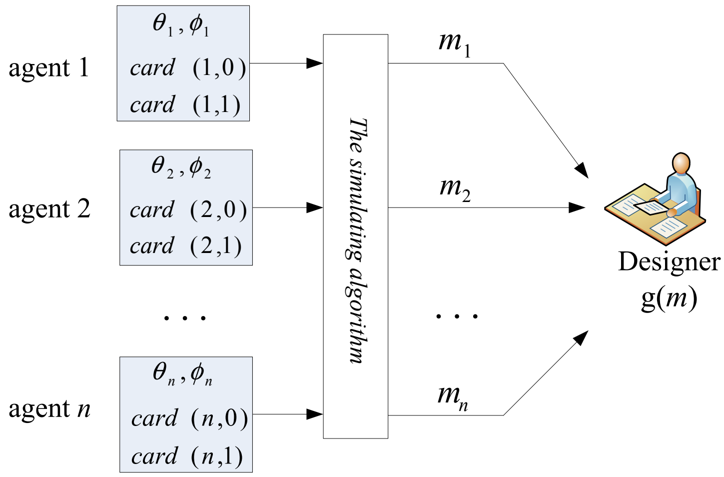

Fig. 2. The inputs and outputs of the simulating algorithm.

$A \times \mathcal{T} \times \mathbb{Z}_+$.

Procedures of the simulating algorithm:
Step 1: Reading two parameters θ_i and ϕ_i from each agent $i \in N$ (See Fig. 3(a)).
Step 2: Computing the leftmost and rightmost columns of $\hat{\omega}_1 \otimes \hat{\omega}_2 \otimes \cdots \otimes \hat{\omega}_n$ (See Fig. 3(b)).
Step 3: Computing the vector representation of $|\psi_2\rangle = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n] \hat{J}_{\pi/2} |C \cdots CC\rangle$.
Step 4: Computing the vector representation of $|\psi_3\rangle = \hat{J}^{+}_{\pi/2} |\psi_2\rangle$.
Step 5: Computing the probability distribution $\langle\psi_3|\psi_3\rangle$ (See Fig. 3(c)).
Step 6: Randomly choosing a "collapsed" state from the set of all 2^n possible states $\{|C \cdots CC\rangle, \cdots, |D \cdots DD\rangle\}$ according to the probability distribution $\langle\psi_3|\psi_3\rangle$.
Step 7: For each $i \in N$, the algorithm sends $card(i, 0)$ (or $card(i, 1)$) as a message m_i to the designer if the i-th basis vector of the "collapsed" state is $|C\rangle$ (or $|D\rangle$) (See Fig. 3(d)).

Remark 1: In Step 6, the possible states $\{|C \cdots CC\rangle, \cdots, |D \cdots DD\rangle\}$ are simply mathematical notions, not physical entities.

Remark 2: Although the time and space complexity of the simulating algorithm are exponential, i.e., $O(2^n)$, it works well when the number of agents is not very large (e.g., less than 20). For example, the runtime of the simulating algorithm is about 0.5s for 15 agents, and about 12s for 20 agents (MATLAB 7.1, CPU: Intel (R) 2GHz, RAM: 3GB).

3.3 An algorithmic version of the quantum mechanism

In the quantum mechanism $\Gamma^Q = ((\hat{S}_i)_{i\in N}, \hat{G})$, the key parts are quantum operations and measurements, which are restricted by current experimental technologies. In Section 3.2, these parts are replaced by a simulating algorithm which can be easily run in a computer. Consequently, the quantum mechanism $\Gamma^Q = ((\hat{S}_i)_{i\in N}, \hat{G})$ shall be updated to an *algorithmic mechanism* $\widetilde{\Gamma} = ((\widetilde{S}_i)_{i\in N}, \widetilde{G})$, which describes a strategy set $\widetilde{S}_i = [0,\pi] \times [0,\pi/2] \times C_i$ for each agent i and an outcome function $\widetilde{G} : [0,\pi]^n \times [0,\pi/2]^n \times \prod_{i\in N} C_i \to A$. We use $\widetilde{S}_{-i}$ to express $[0,\pi]^{n-1} \times [0,\pi/2]^{n-1} \times \prod_{j\neq i} C_j$, and thus, a strategy profile is $\widetilde{s} = (\widetilde{s}_i, \widetilde{s}_{-i})$, where $\widetilde{s}_i = (\theta_i, \phi_i, c_i) \in \widetilde{S}_i$ and $\widetilde{s}_{-i} = (\theta_{-i}, \phi_{-i}, c_{-i}) \in \widetilde{S}_{-i}$. A *Nash equilibrium* of an algorithmic mechanism $\widetilde{\Gamma}$ played in state t is a strategy profile $\widetilde{s}^* = (\widetilde{s}_1^*, \cdots, \widetilde{s}_n^*)$ such that for any agent $i \in N$, $\widetilde{s}_i \in \widetilde{S}_i$, $\widetilde{G}(\widetilde{s}_1^*, \cdots, \widetilde{s}_n^*) \succeq_i^t \widetilde{G}(\widetilde{s}_i, \widetilde{s}_{-i}^*)$.

Working steps of the algorithmic mechanism $\widetilde{\Gamma}$:

Step 1: Given an SCR F and a state t, if two following conditions are satisfied, goto Step 3:
1) There exists $\hat{t} \in \mathcal{T}$, $\hat{t} \neq t$ such that $\hat{a} \succeq_i^t a$ (where $\hat{a} \in F(\hat{t})$, $a \in F(t)$) for every $i \in N$, and $\hat{a} \succ_j^t a$ for at least one $j \in N$;
2) If there exists $\hat{t}' \in \mathcal{T}$, $\hat{t}' \neq \hat{t}$ that satisfies the former condition, then $\hat{a} \succeq_i^t \hat{a}'$ (where $\hat{a} \in F(\hat{t})$, $\hat{a}' \in F(\hat{t}')$) for every $i \in N$, and $\hat{a} \succ_j^t \hat{a}'$ for at least one $j \in N$.
Step 2: Each agent i sets $card(i,0) = (a_i, t_i, z_i)$, and sends $card(i,0)$ as the message m_i to the designer. Goto Step 5.
Step 3: Each agent i sets $card(i,0) = (\hat{a}, \hat{t}, 0)$ and $card(i,1) = (a_i, t_i, z_i)$, then submits θ_i, ϕ_i, $card(i,0)$ and $card(i,1)$ to the simulating algorithm.
Step 4: The simulating algorithm runs in a computer and outputs messages $m_1, \cdots, m_n$ to the designer.
Step 5: The designer receives the overall message $m = (m_1, \cdots, m_n)$ and let the final outcome be $g(m)$ using rules (1)-(3) of the Maskin's mechanism. END.

3.4 Amending sufficient conditions for Nash implementation

As shown in Ref. [2], in the quantum world the sufficient conditions for Nash implementation are amended by virtue of a quantum mechanism. This result looks irrelevant to the macro world because currently the experimental technologies are not commercially available, and people usually feel quantum mechanics is far from macro disciplines such as economics. Here we will show that by using the aforementioned algorithmic mechanism, the sufficient conditions for Nash implementation can be amended immediately just in the macro world.

Following Ref. [2], given n ($n \geq 3$) agents, let us consider the payoff to the n-th agent. We denote by $\$_{C\cdots CC}$ the payoff when all agents submit $\theta = \phi = 0$ in Step 3 of $\widetilde{\Gamma}$ (the "collapsed" state chosen in Step 6 of the simulating algorithm is $|C\cdots CC\rangle$). We denote by $\$_{C\cdots CD}$ the payoff when the first $n-1$ agents choose $\theta = \phi = 0$ and the n-th agent chooses $\theta_n = \pi$, $\phi_n = \pi/n$ (the corresponding "collapsed" state is $|C\cdots CD\rangle$). Note that here $|C\cdots CC\rangle$ and $|C\cdots CD\rangle$ are simply mathematical notions. $\$_{D\cdots DD}$ and $\$_{D\cdots DC}$ are defined similarly.

Now we define condition $\lambda^{\pi/2}$ as follows:
1) $\lambda_1^{\pi/2}$: Given an SCR F and a state t, there exists $\hat{t} \in \mathcal{T}$, $\hat{t} \neq t$ such that $\hat{a} \succeq_i^t a$ (where $\hat{a} \in F(\hat{t})$, $a \in F(t)$) for every $i \in N$, $\hat{a} \succ_j^t a$ for at least one $j \in N$, and the number of agents that encounter a preference change around $\hat{a}$ in going from state $\hat{t}$ to t is at least two. Denote by l the number of these agents. Without loss of generality, let these l agents be the last l agents among n agents.
2) $\lambda_2^{\pi/2}$: If there exists $\hat{t}' \in \mathcal{T}$, $\hat{t}' \neq \hat{t}$ that satisfies $\lambda_1^{\pi/2}$, then $\hat{a} \succeq_i^t \hat{a}'$ (where $\hat{a} \in F(\hat{t})$, $\hat{a}' \in F(\hat{t}')$) for every $i \in N$, and $\hat{a} \succ_j^t \hat{a}'$ for at least one $j \in N$.
3) $\lambda_3^{\pi/2}$: Consider the payoff to the n-th agent, $\$_{C\cdots CC} > \$_{D\cdots DD}$, i.e., he/she prefers the payoff of a certain outcome (generated by rule 1 of the Maskin's mechanism) to the payoff of an uncertain outcome (generated by rule 3 of the Maskin's mechanism).
4) $\lambda_4^{\pi/2}$: Consider the payoff to the n-th agent, $\$_{C\cdots CC} > \$_{C\cdots CD}\cos^2(\pi/l) + \$_{D\cdots DC}\sin^2(\pi/l)$.

Proposition 1: For $n \geq 3$, given a state t and an SCR F that is monotonic and satisfies no-veto:
1) If condition $\lambda^{\pi/2}$ is satisfied, then F is not Nash implementable.
2) If condition $\lambda^{\pi/2}$ is not satisfied (or put differently, condition no-$\lambda^{\pi/2}$ is satisfied), then F is Nash implementable. Thus, the sufficient conditions for Nash implementation are amended as monotonicity, no-veto and no-$\lambda^{\pi/2}$.

Proof: 1) Given a state t and an SCR F, since condition $\lambda_1^{\pi/2}$ and $\lambda_2^{\pi/2}$ are satisfied, then the two conditions in Step 1 of $\widetilde{\Gamma}$ are also satisfied. Hence, the mechanism $\widetilde{\Gamma}$ enters Step 3, i.e., each agent i sets $c_i = (card(i,0), card(i,1)) = ((\hat{a}, \hat{t}, 0), (a_i, t_i, z_i))$, then submits θ_i, ϕ_i, $card(i,0)$ and $card(i,1)$ to the algorithm. Let $c = (c_1, \cdots, c_n)$.

Since condition $\lambda_3^{\pi/2}$ and $\lambda_4^{\pi/2}$ are satisfied, then according to Proposition 2 in Ref. [2], if the n agents choose $\widetilde{s}^* = (\theta^*, \phi^*, c)$, where $\theta^* = (\underbrace{0, \cdots, 0}_{n})$,

$\phi^* = (\underbrace{0, \cdots, 0}_{n-l}, \underbrace{\pi/l, \cdots, \pi/l}_{l})$, then $\widetilde{s}^* \in \mathcal{N}(\widetilde{\Gamma}, t)$. In Step 6 of the simulating algorithm, the chosen "collapsed" state is $|C \cdots CC\rangle$. Hence, in Step 7 of the simulating algorithm, $m_i = card(i, 0) = (\hat{a}, \hat{t}, 0)$ for each agent $i \in N$. Finally, in Step 5 of $\widetilde{\Gamma}$, $\widetilde{G}(\widetilde{s}^*) = g(m) = \hat{a} \notin F(t)$. Hence, F is not Nash implementable.

2) If condition $\lambda^{\pi/2}$ is not satisfied, then no matter whether $\widetilde{\Gamma}$ enters Step 2 or Step 3, the aforementioned novel Nash equilibrium which yields the Pareto-efficient outcome $\hat{a}$ will no longer exist. Hence, $\mathcal{N}(\widetilde{\Gamma}, t) = \mathcal{N}(\Gamma, t)$ for every $t \in \mathcal{T}$, where Γ is the traditional Maskin's mechanism. Since the SCR F is monotonic and satisfies no-veto, then it is Nash implementable. $\square$

3.5 Discussions

Remark 3: Just like what we have seen in the quantum mechanism, the algorithmic mechanism does not mean that the agents have the designer to design a new mechanism to attain a Pareto superior outcome. From the designer's perspective, there is no difference between the algorithmic mechanism and the traditional Maskin's mechanism. As a comparison, the strategy space of each agent has been enlarged in the algorithmic mechanism.

Remark 4: See Fig. 3(b) and Fig. 3(c), the introduction of complex numbers is a novel idea to the theory of mechanism design. To the best of our knowledge, up to now there is no similar work before. Indeed, this introduction is indispensable for the amendment of sufficient conditions of the Maskin's theorem, because only by using complex numbers can quantum properties be simulated in a computer.

Remark 5: Although the algorithmic mechanism uses complex numbers in the simulating algorithm, it is a completely *classical* mechanism that can be run in a computer. In addition, condition $\lambda^{\pi/2}$ is also a classical condition. Therefore, the sufficient conditions for Nash implementation are amended immediately in the classical macro world (Note: here the phrase "macro world" only stands for the computer or Internet world, where the algorithmic mechanism is meaningful).

Remark 6: The problem of Nash implementation requires complete information among all agents. In the last paragraph of Page 392, Ref. [4], Serrano wrote: "*We assume that there is complete information among the agents... This assumption is especially justified when the implementation problem concerns a small number of agents that hold good information about one another*". Hence, the fact that the algorithmic mechanism is suitable for small-scale cases (e.g., less than 20 agents) is acceptable for Nash implementation.

4 Conclusions

Just like quantum mechanics brings novel results to physics, quantum computing leads new ideas to game theory [5,6]. By coincidence, Maskin [1] and Eisert *et al* [5] formally published their papers in the same year 1999. The two disciplines, mechanism design and quantum games, were not connected until the theory of mechanism design was generalized to the quantum domain in 2011 [2]. In this paper, we go beyond the obstacle of how to realize the quantum mechanism, and propose an algorithmic mechanism which amends the sufficient conditions for Nash implementation in the macro world. Since the Maskin's mechanism has been widely applied among the mechanism design literature, there are many works to do in the future to generalize the algorithmic mechanism.

References

[1] E. Maskin, Nash equilibrium and welfare optimality, *Rev. Econom. Stud.* **66** (1999) 23-38.

[2] H. Wu, Quantum mechanism helps agents combat "bad" social choice rules. *International Journal of Quantum Information* **9** (2011) 615-623. http://arxiv.org/abs/1002.4294

[3] T.D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe and J.L. O'Brien, Quantum computers, *Nature*, **464** (2010) 45-53.

[4] R. Serrano, The theory of implementation of social choice rules, *SIAM Review* **46** (2004) 377-414.

[5] J. Eisert, M. Wilkens and M. Lewenstein, Quantum games and quantum strategies, *Phys. Rev. Lett.* **83** (1999) 3077-3080.

[6] A.P. Flitney and L.C.L. Hollenberg, Nash equilibria in quantum games with generalized two-parameter strategies, *Phys. Lett. A* **363** (2007) 381-388.

```
start_time = cputime

% n: the number of agents. In Example 1 of Ref. [2], there are 3 agents: Apple, Lily, Cindy
n=3;

% gamma: the coefficient of entanglement. Here we simply set gamma to its maximum π/2.
gamma=pi/2;

% Defining the array of θ_i and φ_i, i = 1,⋯,n.
theta=zeros(n,1);
phi=zeros(n,1);

% Reading Apple's parameters. For example, ω̂_1 = Ĉ_2 = ω̂(0,π/2)
theta(1)=0;
phi(1)=pi/2;

% Reading Lily's parameters. For example, ω̂_2 = Ĉ_2 = ω̂(0,π/2)
theta(2)=0;
phi(2)=pi/2;

% Reading Cindy's parameters. For example, ω̂_3 = Î = ω̂(0,0)
theta(3)=0;
phi(3)=0;
```

Fig. 3 (a). Reading each agent *i*'s parameters θ_i and $\phi_i, i = 1, \cdots, n$.

```
% Defining two 2*2 matrices
A=zeros(2,2);
B=zeros(2,2);

% In the beginning, A represents the local operation ω̂_1 of agent 1. (See Eq 7)
A(1,1)=exp(i*phi(1))*cos(theta(1)/2);
A(1,2)=i*sin(theta(1)/2);
A(2,1)=A(1,2);
A(2,2)=exp(-i*phi(1))*cos(theta(1)/2);
row_A=2;

% Computing ω̂_1 ⊗ ω̂_2 ⊗ ⋯ ⊗ ω̂_n
for agent=2 : n
        % B varies from ω̂_2 to ω̂_n
        B(1,1)=exp(i*phi(agent))*cos(theta(agent)/2);
        B(1,2)=i*sin(theta(agent)/2);
        B(2,1)=B(1,2);
        B(2,2)=exp(-i*phi(agent))*cos(theta(agent)/2);

        % Computing the leftmost and rightmost columns of C= A ⊗ B
        C=zeros(row_A*2, 2);
        for row=1 : row_A
                C((row-1)*2+1, 1) = A(row,1) * B(1,1);
                C((row-1)*2+2, 1) = A(row,1) * B(2,1);
                C((row-1)*2+1, 2) = A(row,2) * B(1,2);
                C((row-1)*2+2, 2) = A(row,2) * B(2,2);
        end
        A=C;
        row_A = 2 * row_A;
end
% Now the matrix A contains the leftmost and rightmost columns of ω̂_1 ⊗ ω̂_2 ⊗ ⋯ ⊗ ω̂_n
```

Fig. 3 (b). Computing the leftmost and rightmost columns of $\hat{\omega}_1 \otimes \hat{\omega}_2 \otimes \cdots \otimes \hat{\omega}_n$

```
% Computing |ψ2⟩ = [ω̂1 ⊗ ω̂2 ⊗ ··· ⊗ ω̂n] Ĵ |C···CC⟩
psi2=zeros(power(2,n),1);
for row=1 : power(2,n)
        psi2(row)=A(row,1)*cos(gamma/2)+A(row,2)*i*sin(gamma/2);
end

% Computing |ψ3⟩ = Ĵ+ |ψ2⟩
psi3=zeros(power(2,n),1);
for row=1 : power(2,n)
        psi3(row)=cos(gamma/2)*psi2(row) - i*sin(gamma/2)*psi2(power(2,n)-row+1);
end

% Computing the probability distribution ⟨ψ3|ψ3⟩
distribution=psi3.*conj(psi3);
distribution=distribution./sum(distribution);
```

Fig. 3 (c). Computing $|\psi_2\rangle, |\psi_3\rangle, \langle\psi_3|\psi_3\rangle$.

```
temp=0;
for index=1: power(2,n)
   temp = temp + distribution(index);
   if temp >= random_number
      break;
   end
end

% indexstr: a binary representation of the index of the collapsed state
%   '0' stands for |C⟩, '1' stands for |D⟩
indexstr=dec2bin(index-1);
sizeofindexstr=size(indexstr);

% Defining an array of messages for all agents
message=cell(n,1);

% For each agent i ∈ N, the algorithm generates the message m_i
for index=1 : n - sizeofindexstr(2)
   message{index,1}=strcat('card(',int2str(index),',0)');
end
for index=1 : sizeofindexstr(2)
   if indexstr(index)=='0'      % Note: '0' stands for |C⟩
      message{n-sizeofindexstr(2)+index,1}=strcat('card(',int2str(n-sizeofindexstr(2)+index),',0)');
   else
      message{n-sizeofindexstr(2)+index,1}=strcat('card(',int2str(n-sizeofindexstr(2)+index),',1)');
   end
end

% The algorithm sends messages m_1, m_2, ···, m_n to the designer
for index=1:n
   disp(message(index));
end

end_time = cputime;
runtime=end_time – start_time
```

Fig. 3 (d). Computing all messages $m_1, m_2, \cdots, m_n$.