Quantum Bayesian implementation

Wu, Haoyang

28 April 2011

Quantum Bayesian implementation

Abstract

Bayesian implementation concerns decision making problems when agents have incomplete information. A recent work [Wu, Quantum mechanism helps agents combat "bad" social choice rules. *Intl. J. of Quantum Information* 9 (2011) 615-623] generalized the implementation theory with complete information to a quantum domain. In this paper, we propose a quantum Bayesian mechanism and an algorithmic Bayesian mechanism, which amend the traditional results for Bayesian implementation.

Key words: Algorithmic mechanism design; Bayesian implementation.

1 Introduction

Mechanism design is an important branch of economics. Compared with game theory, it concerns a reverse question: given some desirable outcomes, can we design a game that produces them? Nash implementation and Bayesian implementation are two key topics of the mechanism design theory. The former assumes complete information among the agents, whereas the latter concerns incomplete information. Maskin [1] provided an almost complete characterization of social choice rules that are Nash implementable when the number of agents is at least three. Postlewaite and Schmeidler [2], Palfrey and Srivastava [3], and Jackson [4] together constructed a framework for Bayesian implementation.

In 2011, Wu [5] claimed that the sufficient conditions for Nash implementation shall be amended by virtue of a quantum mechanism. Furthermore, this amendment holds in the macro world by virtue of an algorithmic mechanism [6]. Given these accomplishments in the field of Nash implementation, this paper aims to investigate what will happen if the quantum mechanism is applied to Bayesian implementation.

The rest of this paper is organized as follows: Section 2 recalls preliminaries of Bayesian implementation given by Serrano [7]. In Section 3, a novel condition, multi-Bayesian monotonicity, is defined. Section 4 and 5 are the main parts of this paper, in which we will propose quantum and algorithmic Bayesian mechanisms respectively. The last section draws the conclusions.

2 Preliminaries

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*. Consider environments in which the state $t = (t_1, \cdots, t_n)$ is not common knowledge among the n agents. We denote by T the set of states compatible with an environment, i.e., a set of states that is common knowledge among the agents. Let $T = \prod_{i \in N} T_i$. Each agent $i \in N$ knows his type $t_i \in T_i$, but not necessarily the types of the others. We will use the notation t_{-i} to denote $(t_j)_{j \neq i}$. Similarly, $T_{-i} = \prod_{j \neq i} T_j$.

Each agent has a *prior belief*, probability distribution, q_i defined on T. We make an assumption of nonredundant types: for every $i \in N$ and $t_i \in T_i$, there exists $t_{-i} \in T_{-i}$ such that $q_i(t) > 0$. For each $i \in N$ and $t_i \in T_i$, the conditional probability of $t_{-i} \in T_{-i}$, given t_i, is the *posterior belief* of type t_i and it is denoted $q_i(t_{-i}|t_i)$. For simplicity, we shall consider only single-valued rules, i.e., an SCF f is a mapping $f : T \mapsto A$. Let $\mathcal{F}$ denote the set of SCFs. Given agent i's state t_i and utility function $u_i(\cdot, t) : \Delta \times T \mapsto \mathbb{R}$, the *conditional expected utility* of agent i of type t_i corresponding to a social choice function (SCF) $f : T \mapsto \Delta$ is defined as:

$$U_i(f|t_i) \equiv \sum_{t'_{-i} \in T_{-i}} q_i(t'_{-i}|t_i) u_i(f(t'_{-i}, t_i), (t'_{-i}, t_i)).$$

An *environment with incomplete information* is a list $E =< N, A, (u_i, T_i, q_i)_{i \in N} >$. An environment is *economic* if, as part of the social outcomes, there exists a private good (e.g., money) over which all agents have a strictly positive preference. Two SCFs f and h are *equivalent* ($f \approx h$) if $f(t) = h(t)$ for every $t \in T$.

Consider a *mechanism* $\Gamma = ((M_i)_{i \in N}, g)$ imposed on an incomplete information environment E, $g : M \mapsto \mathcal{F}$. A *Bayesian Nash equilibrium* of Γ is a profile of strategies $\sigma^* = (\sigma^*_i)_{i \in N}$ where $\sigma^*_i : T_i \mapsto M_i$ such that for all $i \in N$ and for all $t_i \in T_i$,

$$U_i(g(\sigma^*)|t_i) \geq U_i(g(\sigma^*_{-i}, \sigma'_i)|t_i), \quad \forall \sigma'_i : T_i \mapsto M_i.$$

Denote by $\mathcal{B}(\Gamma)$ the set of Bayesian equilibria of the mechanism Γ. Let $g(\mathcal{B}(\Gamma))$ be the corresponding set of equilibrium outcomes. An SCF f is *Bayesian implementable* if there exists a mechanism $\Gamma = ((M_i)_{i \in N}, g)$ such that $g(\mathcal{B}(\Gamma)) \approx f$. An SCF f is *incentive compatible* if truth-telling is a Bayesian equilibrium of the direct mechanism associated with f, i.e., if for every $i \in N$ and for every $t_i \in T_i$,

$$\sum_{t'_{-i} \in T_{-i}} q_i(t'_{-i}|t_i) u_i(f(t'_{-i}, t_i), (t'_{-i}, t_i)) \geq \sum_{t'_{-i} \in T_{-i}} q_i(t'_{-i}|t_i) u_i(f(t'_{-i}, t'_i), (t'_{-i}, t_i)),$$

$\forall t'_i \in T_i$.

Consider a strategy in a direct mechanism for agent i, i.e., a mapping $\alpha_i = (\alpha_i(t_i))_{t_i \in T_i} : T_i \mapsto T_i$. A *deception* $\alpha = (\alpha_i)_{i\in N}$ is a collection of such mappings where at least one differs from the identity mapping. Given an SCF f and a deception α, let $[f \circ \alpha]$ denote the following SCF: $[f \circ \alpha](t) = f(\alpha(t))$ for every $t \in T$. For a type $t_i \in T_i$, an SCF f, and a deception α, let $f_{\alpha_i(t_i)}(t') = f(t'_{-i}, \alpha_i(t_i))$ for all $t' \in T$. An SCF f is *Bayesian monotonic* if for any deception α, whenever $f \circ \alpha \not\approx f$, there exist $i \in N$, $t_i \in T_i$, and an SCF y such that

$$U_i(y \circ \alpha|t_i) > U_i(f \circ \alpha|t_i), \quad \text{while } U_i(y_{\alpha_i(t_i)}|t'_i) \leq U_i(f|t'_i), \quad \forall t'_i \in T_i. \quad (*).$$

In economic environments, the sufficient and necessary conditions for full Bayesian implementation are incentive compatibility and Bayesian monotonicity. To facilitate the following discussion, here we cite the Bayesian mechanism (Page 404, line 4, [7]) as follows: Consider a mechanism $\Gamma = ((M_i)_{i\in N}, g)$, where $M_i = T_i \times \mathcal{F} \times \mathbb{Z}_+$, and $\mathbb{Z}_+$ is the set of nonnegative integers. Each agent is asked to report his type t_i, an SCF f_i and a nonnegative integer z_i, i.e., $m_i = (t_i, f_i, z_i)$. The outcome function g is as follows:
(i) If for all $i \in N$, $m_i = (t_i, f, 0)$, then $g(m) = f(t)$, where $t = (t_1, \cdots, t_n)$.
(ii) If for all $j \neq i$, $m_j = (t_j, f, 0)$ and $m_i = (t'_i, y, z_i) \neq (t'_i, f, 0)$, we can have two cases:
(a) If for all t_i, $U_i(y_{t'_i}|t_i) \leq U_i(f|t_i)$, then $g(m) = y(t'_i, t_{-i})$;
(b) Otherwise, $g(m) = f(t'_i, t_{-i})$.
(iii) In all other cases, the total endowment of the economy is awarded to the agent of smallest index among those who announce the largest integer.

3 Multi-Bayesian monotonicity

Definition 1: An SCF f is *multi-Bayesian monotonic* if there exist a deception α, $f \circ \alpha \not\approx f$, and a set of agents $N^\alpha = \{i^1, i^2, \cdots\} \subseteq N$, $2 \leq |N^\alpha| \leq n$, such that for every $i \in N^\alpha$, there exist $t_i \in T_i$ and an SCF $y^i \in \mathcal{F}$ that satisfy:

$$U_i(y^i \circ \alpha|t_i) > U_i(f \circ \alpha|t_i), \quad \text{while } U_i(y^i_{\alpha_i(t_i)}|t'_i) \leq U_i(f|t'_i), \quad \forall t'_i \in T_i. \quad (**).$$

Let $l = |N^\alpha|$. Without loss of generality, let these l agents be the last l agents among n agents.

In 1993, Matsushima [9] claimed that Bayesian monotonicity is a very weak condition when utility functions are quasi-linear and lotteries are available. Consider an SCF f that satisfies Bayesian monononicity, if there is a deception α such that its corresponding agent i has another symmetric agent j (i.e., $i \neq j$, $u_i = u_j$, $T_i = T_j$, the prior belief and posterior belief hold by them are the same), then f is multi-Bayesian monotonic.

Example 1: Similar to Example 23.B.5 in Ref. [8], here we consider an auction setting with one seller (i.e., agent 0) and three buyers (i.e., agent 1, 2 and 3). All buyers' privately observed valuations t_i are drawn independently from the uniform distribution on $[0, 1]$ and this fact is common knowledge among the agents. Each buyer submits a sealed bid, $b_i \geq 0$ ($i = 1, 2, 3$). The sealed bids are examined and the buyer with the highest bid is declared the winner. If there is a tie, the winner is chosen randomly. The winning buyer pays an amount equal to his bid to the seller. The losing buyer does not pay anything.

Consider the social choice function $f(t) = (x_0(t), x_1(t), x_2(t), x_3(t), p_0(t), p_1(t), p_2(t), p_3(t))$, in which

$$\begin{aligned}
x_1(t) &= 1, \quad \text{if } t_1 \geq t_2 \text{ and } t_1 \geq t_3; \quad = 0 \text{ otherwise;}\\
x_2(t) &= 1, \quad \text{if } t_2 > t_1 \text{ and } t_2 \geq t_3; \quad = 0 \text{ otherwise;}\\
x_3(t) &= 1, \quad \text{if } t_3 > t_1 \text{ and } t_3 > t_2; \quad = 0 \text{ otherwise;}\\
x_0(t) &= 0, \quad \text{for all } t;\\
p_1(t) &= -\frac{2}{3}\theta_1 x_1(t);\\
p_2(t) &= -\frac{2}{3}\theta_2 x_2(t);\\
p_3(t) &= -\frac{2}{3}\theta_3 x_3(t);\\
p_0(t) &= -[p_1(t) + p_2(t) + p_3(t)].
\end{aligned}$$

It can be easily checked that the strategies $b_i(t_i) = \frac{2}{3}t_i$ (for $i = 1, 2, 3$) constitute a Bayesian Nash equilibrium of this auction that indirectly yields the outcomes specified by $f(t)$. Thus, according to Theorem 1 [4], f is incentive compatible and Bayesian monotonic. Since the three buyers are symmetric, then according to the definition of multi-Bayesian monotonicity, f is multi-Bayesian monotonic.

Proposition 1: In economic environments, consider an SCF f that is incentive compatible and Bayesian monotonic, if f is multi-Bayesian monotonic, then $f \circ \alpha$ is not Bayesian implementable by using the traditional Bayesian mechanism, where α is specified in the definition of multi-Bayesian monotonicity.

Proof: According to Serrano's proof (Page 404, line 33, [7]), all equilibrium strategies fall under rule (i), i.e., f is unanimously announced and all agents announce the integer 0. Consider the deception α specified in the definition of multi-Bayesian monotonicity. At first sight, if every agent $i \in N$ submits $(\alpha_i(t_i), f, 0)$, then $f \circ \alpha$ may be generated as the equilibrium outcome by rule (i). However, For each agent $i \in N^\alpha$, he has incentives to unilaterally deviate from $(\alpha_i(t_i), f, 0)$ to $(\alpha_i(t_i), y^i, 0)$ in order to obtain $y^i \circ \alpha$ by rule (ii.a). This is a profitable deviation for each agent $i \in N^\alpha$. Therefore, $f \circ \alpha$ is not Bayesian implementable. □

4 A quantum Bayesian mechanism

Following Ref. [5], here we will propose a quantum Bayesian mechanism to modify the sufficient conditions for Bayesian implementation. According to Eq (4) in Ref. [10], 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, σ_x is Pauli matrix), $\hat{I} \equiv \hat{\omega}(0,0)$, $\hat{D}_n \equiv \hat{\omega}(\pi,\pi/n)$, $\hat{C}_n \equiv \hat{\omega}(0,\pi/n)$.

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 information 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 $c_i = (card(i,0), card(i,1))$, where $card(i,0), card(i,1) \in T_i \times \mathcal{F} \times \mathbb{Z}_+$. 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 Bayesian mechanism* $\Gamma_B^Q = ((\hat{\Sigma}_i)_{i\in N}, \hat{g})$ describes a strategy set $\hat{\Sigma}_i = \{\hat{\sigma}_i : T_i \mapsto \hat{\Omega}_i \times C_i\}$ for each agent i and an outcome function $\hat{g} : \otimes_{i\in N}\hat{\Omega}_i \times \prod_{i\in N} C_i \mapsto \mathcal{F}$. A strategy profile is $\hat{\sigma} = (\hat{\sigma}_i, \hat{\sigma}_{-i})$, where $\hat{\sigma}_{-i} : T_{-i} \mapsto \otimes_{j\neq i}\hat{\Omega}_j \times \prod_{j\neq i} C_j$. A *Bayesian Nash equilibrium* of Γ_B^Q is a strategy profile $\hat{\sigma}^* = (\hat{\sigma}_1^*, \cdots, \hat{\sigma}_n^*)$ such that for every $i \in N$ and for every $t_i \in T_i$,

$$U_i(\hat{g}(\hat{\sigma}^*)|t_i) \geq U_i(\hat{g}(\hat{\sigma}_{-i}^*, \hat{\sigma}_i')|t_i), \quad \forall \hat{\sigma}_i' : T_i \mapsto \hat{\Omega}_i \times C_i.$$

The setup of the quantum Bayesian mechanism $\Gamma_B^Q = ((\hat{\Sigma}_i)_{i\in N}, \hat{g})$ is depicted in Fig. 1. The working steps of Γ_B^Q are given as follows:
Step 1: Nature selects a state $t \in T$ and assigns t to the agents. Each agent i knows t_i and $q_i(t_{-i}|t_i)$. The state of each 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: If f is multi-Bayesian monotonic, then go to Step 4.

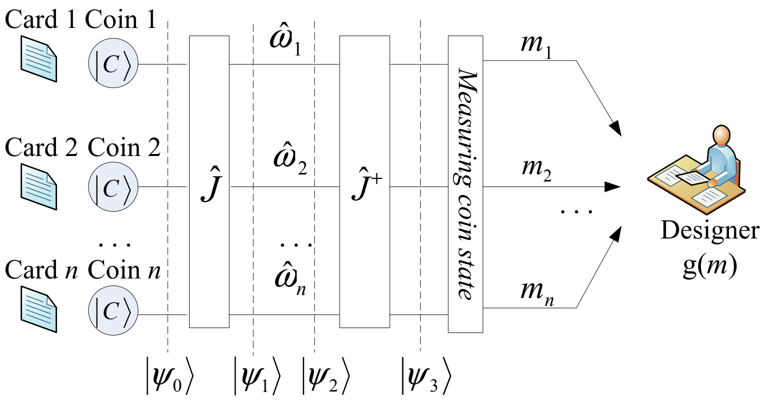

Fig. 1. The setup of a quantum Bayesian mechanism. Each agent has a quantum coin and a card. Each agent independently performs a local unitary operation on his/her own quantum coin.

Step 3: Each agent i sets $c_i = ((t_i, f_i, z_i), (t_i, f_i, z_i))$, $\hat{\omega}_i = \hat{I}$. Go to Step 7.
Step 4: Each agent i sets $c_i = ((\alpha_i(t_i), f, 0), (t_i, f_i, z_i))$ (where α is specified in the definition of multi-Bayesian monotonicity). Let n quantum coins be entangled by $\hat{J}$. $|\psi_1\rangle = \hat{J}|\psi_0\rangle$.
Step 5: Each agent i independently performs a local unitary operation $\hat{\omega}_i$ on his/her own quantum coin. $|\psi_2\rangle = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]|\psi_1\rangle$.
Step 6: Let n quantum coins be disentangled by $\hat{J}^+$. $|\psi_3\rangle = \hat{J}^+|\psi_2\rangle$.
Step 7: The device measures the state of n quantum coins and sends $card(i, 0)$ (or $card(i, 1)$) as 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 $\hat{g}(\hat{\sigma}) = g(m)$ using rules (i)-(iii) specified in the traditional Bayesian mechanism. END.

Given $n \geq 3$ agents and an SCF f, suppose f satisfies multi-Bayesian monotonicity. For each $i \in N^\alpha$, let $card(i, 0) = (\alpha_i(t_i), f, 0)$, $card(i, 1) = (\alpha_i(t_i), y^i, 0)$; for each $i \notin N^\alpha$, let $card(i, 0) = (\alpha_i(t_i), f, 0)$, $card(i, 1) = (t_i, f_i, z_i)$ (where α, N^α, y^i are specified in the definition of multi-Bayesian monotonicity). We define the payoff to the n-th agent as follows: $\$_{C\cdots CC}$ represents the payoff to the n-th agent when the measured state of n quantum coins in Step 7 of Γ_B^Q is $|\underbrace{C\cdots CC}_{n}\rangle$; $\$_{C\cdots CD}$ represents the payoff to the n-th agent when the measured state of n quantum coins is $|\underbrace{C\cdots C}_{n-1}D\rangle$). $\$_{D\cdots DD}$ and $\$_{D\cdots DC}$ are defined similarly.

Definition 2: Given an SCF f satisfying multi-Bayesian monotonicity, define condition λ^B as follows:
1) λ_1^B: Consider the payoff to the n-th agent, $\$_{C\cdots CC} > \$_{D\cdots DD}$, i.e., he/she prefers the expected payoff of a certain outcome (generated by rule (i)) to the expected payoff of an uncertain outcome (generated by rule (iii)).
2) λ_2^B: Consider the payoff to the n-th agent, $\$_{C\cdots CC} > \$_{C\cdots CD}[1 - \sin^2\gamma \sin^2(\pi/l)] + \$_{D\cdots DC} \sin^2\gamma \sin^2(\pi/l)$.

Proposition 2: In economic environments, consider an SCF f that is incentive compatible and Bayesian monotonic, if f is multi-Bayesian monotonic and condition λ^B is satisfied, then $f \circ \alpha$ is Bayesian implementable by using the quantum Bayesian mechanism.
Proof: Since f is multi-Bayesian monotonic, then there exist a deception α, $f \circ \alpha \not\approx f$, and $2 \le l \le n$ agents that satisfy Eq (**), i.e., for each agent $i \in N^\alpha$, there exist $t_i \in T_i$ and an SCF $y^i \in \mathcal{F}$ such that:

$$U_i(y^i \circ \alpha | t_i) > U_i(f \circ \alpha | t_i), \quad \text{while } U_i(y^i_{\alpha_i(t_i)} | t_i') \le U_i(f | t_i'), \quad \forall t_i' \in T_i.$$

Hence, the quantum Bayesian mechanism will enter Step 4. Each agent $i \in N$ sets $c_i = ((\alpha_i(t_i), f, 0), (t_i, f_i, z_i))$. Let $c = (c_1, \cdots, c_n)$. Since condition λ^B is satisfied, then similar to the proof of Proposition 2 in Ref. [5], if the n agents choose $\hat{\sigma}^* = (\hat{\omega}^*, c)$, where $\hat{\omega}^* = (\underbrace{\hat{I}, \cdots, \hat{I}}_{n-l}, \underbrace{\hat{C}_l, \cdots, \hat{C}_l}_{l})$, then $\hat{\sigma}^* \in \mathcal{B}(\Gamma^Q_B)$. In Step 7, the corresponding measured state of n quantum coins is $|\underbrace{C \cdots CC}_{n}\rangle$. Hence, for each agent $i \in N$, $m_i = (\alpha_i(t_i), f, 0)$. In Step 8, $\hat{g}(\hat{\sigma}^*) = f \circ \alpha \not\approx f$.
Therefore, $f \circ \alpha$ is implemented by Γ^Q_B in Bayesian Nash equilibrium. □

5 An algorithmic Bayesian mechanism

Following Ref. [6], in this section we will propose an algorithmic Bayesian mechanism to help agents benefit from the quantum Bayesian mechanism in the macro world. In the beginning, we cite matrix representations of quantum states from Ref. [6].

5.1 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 $(0,1)^T$. In the beginning, we define:

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

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

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

For $\gamma = \pi/2$,

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

5.2 A simulating algorithm

Similar to Ref. [6], in the following we will propose a simulating algorithm that simulates the quantum operations and measurements in Steps 4-7 of the quantum Bayesian mechanism given in Section 4. The inputs and outputs of the algorithm are adjusted to the case of Bayesian implementation. The factor γ is also set as its maximum $\pi/2$. For n agents, the inputs and outputs of the algorithm are illustrated in Fig. 2. The *Matlab* program is given in Fig. 3, which is cited from Ref. [6].

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 $card(i, 0), card(i, 1) \in T_i \times \mathcal{F} \times \mathbb{Z}_+$.

Outputs:
m_i, $i = 1, \cdots, n$: the agent i's message that is sent to the designer, $m_i \in T_i \times \mathcal{F} \times \mathbb{Z}_+$.

Procedures of the algorithm:
Step 1: Reading 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 $\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n$ (See Fig.

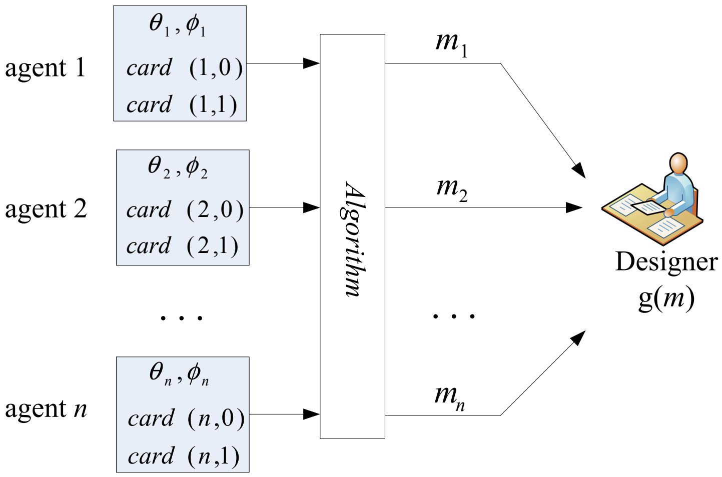

Fig. 2. The inputs and outputs of the algorithm.

3(b)).
Step 3: Computing the vector representation of $|\psi_2\rangle = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}_{\pi/2}|\psi_0\rangle$.
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 $\{\underbrace{|C\cdots CC\rangle}_{n}, \cdots, \underbrace{|D\cdots DD\rangle}_{n}\}$ 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 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)).

5.3 *An algorithmic version of the quantum Bayesian mechanism*

In the quantum Bayesian mechanism $\Gamma^Q_B = ((\hat{\Sigma}_i)_{i\in N}, \hat{g})$, the key parts are quantum operations and measurements, which are restricted by current experimental technologies. In Section 5.2, these parts are replaced by a simulating algorithm which can be easily run in a computer. Now we update the quantum Bayesian mechanism $\Gamma^Q_B = ((\hat{\Sigma}_i)_{i\in N}, \hat{g})$ to an *algorithmic Bayesian mechanism* $\widetilde{\Gamma}_B = ((\widetilde{\Sigma}_i)_{i\in N}, \widetilde{g})$, which describes a strategy set $\widetilde{\Sigma}_i = \{\widetilde{\sigma}_i : T_i \mapsto [0,\pi]\times[0,\pi/2]\times C_i\}$ for each agent i and an outcome function $\widetilde{g} : [0,\pi]^n \times [0,\pi/2]^n \times \prod_{i\in N} C_i \to \mathcal{F}$, where $n \geq 3$, C_i is the set of agent i's card $c_i = (card(i,0), card(i,1))$. A typical message sent by agent i is denoted by $(\theta_i, \phi_i, t_i, f_i, z_i, t'_i, f'_i, z'_i)$.

A strategy profile is $\widetilde{\sigma} = (\widetilde{\sigma}_i, \widetilde{\sigma}_{-i})$, where $\widetilde{\sigma}_{-i} : T_{-i} \mapsto [0,\pi]^{n-1} \times [0,\pi/2]^{n-1} \times \prod_{j\neq i} C_j$. A Bayesian Nash equilibrium of $\widetilde{\Gamma}_B$ is a strategy profile $\widetilde{\sigma}^* = (\widetilde{\sigma}^*_1, \cdots, \widetilde{\sigma}^*_n)$ such that for any agent $i \in N$ and for all $t_i \in T_i$,

$$U_i(\widetilde{g}(\widetilde{\sigma}^*)|t_i) \geq U_i(\widetilde{g}(\widetilde{\sigma}^*_{-i}, \widetilde{\sigma}'_i)|t_i), \quad \forall \widetilde{\sigma}'_i : T_i \mapsto [0,\pi]\times[0,\pi/2]\times C_i.$$

Since the factor γ is set as its maximum $\pi/2$, the condition λ^B in the quantum Bayesian mechanism shall be updated as $\lambda^{B\pi/2}$. $\lambda_1^{B\pi/2}$ is the same as λ_1^B; $\lambda_2^{B\pi/2}$ is revised as follows: Consider the payoff to the n-th agent, $\$_{C\cdots CC} > \$_{C\cdots CD}\cos^2(\pi/l) + \$_{D\cdots DC}\sin^2(\pi/l)$.

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

Step 1: Given an SCF f, if f is multi-Bayesian monotonic, go to Step 3.
Step 2: Each agent i sends (t_i, f_i, z_i) as m_i to the designer. Go to Step 5.
Step 3: Each agent $i \in N^\alpha$ sets $card(i,0) = (\alpha_i(t_i), f, 0)$, $card(i,1) = (\alpha_i(t_i), y^i, 0)$; each agent $i \notin N^\alpha$ sets $card(i,0) = (\alpha_i(t_i), f, 0)$, $card(i,1) = (t_i, f_i, z_i)$ (where α, N^α, y^i are specified in the definition of multi-Bayesian monotonicity). Then each agent i submits θ_i, ϕ_i, $card(i,0)$ and $card(i,1)$ to the simulating algorithm.
Step 4: The simulating algorithm runs 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 (i)-(iii) of the traditional Bayesian mechanism. END.

5.4 New results for Bayesian implementation

Proposition 3: In economic environments, given an SCF f that is incentive compatible and Bayesian monotonic:
1) If f is multi-Bayesian monotonic and condition $\lambda^{B\pi/2}$ is satisfied, then $f \circ \alpha$ is Bayesian implementable by using the algorithmic Bayesian mechanism.
2) If f is not multi-Bayesian monotonic, then f is Bayesian implementable.
Proof: 1) Since f is multi-Bayesian monotonic, then $\widetilde{\Gamma}_B$ enters Step 3.
Each agent $i \in N^\alpha$ sets $card(i,0) = (\alpha_i(t_i), f, 0)$, $card(i,1) = (\alpha_i(t_i), y^i, 0)$; each agent $i \notin N^\alpha$ sets $card(i,0) = (\alpha_i(t_i), f, 0)$, $card(i,1) = (t_i, f_i, z_i)$ (where α, N^α, y^i are specified in the definition of multi-Bayesian monotonicity). Then each agent i submits θ_i, ϕ_i, $card(i,0)$ and $card(i,1)$ to the simulating algorithm. Since condition $\lambda^{B\pi/2}$ is satisfied, then similar to the proof of Proposition 1 in Ref. [6], if the n agents choose $\widetilde{\sigma}^* = (\widetilde{\sigma}_i^*)_{i\in N}$, where for $1 \le i \le (n-l)$, $\theta_i = \phi_i = 0$; for $(n-l+1) \le i \le n$, $\theta_i = 0$, $\phi_i = \pi/l$, then $\widetilde{\sigma}^* \in \mathcal{B}(\widetilde{\Gamma}_B))$.
In Step 6 of the simulating algorithm, the corresponding measured state is $\underbrace{|C\cdots CC\rangle}_{n}$.
Hence, in Step 7 of the simulating algorithm, $m_i = card(i,0) = (\alpha_i(t_i), f, 0)$ for each agent $i \in N$. Finally, in Step 5 of $\widetilde{\Gamma}_B$, $\widetilde{g}(\widetilde{\sigma}^*) = g(m) = f \circ \alpha \not\approx f$.
Therefore, $f \circ \alpha$ is implemented by $\widetilde{\Gamma}_B$ in Bayesian Nash equilibrium.
2) If f is not multi-Bayesian monotonic, then $\widetilde{\Gamma}_B$ is reduced to the traditional Bayesian mechanism. Since the SCF f is incentive compatible and Bayesian monotonic, then it is Bayesian implementable. □

6 Conclusions

This paper follows the series of papers on quantum mechanisms [5,6], and generalizes the quantum and algorithmic mechanisms in Refs. [5,6] to Bayesian implementation. It can be seen that for n agents, the time complexity of quantum and algorithmic Bayesian mechanisms are $O(n)$ and $O(2^n)$ respectively. Although current experimental technologies restrict the quantum Bayesian mechanism to be commercially available, for small-scale cases (e.g., less than 20 agents [6]), the algorithmic Bayesian mechanism can help agents benefit from quantum Bayesian mechanism just in the macro world.

Acknowledgments

The author is very grateful to Ms. Fang Chen, Hanyue Wu (*Apple*), Hanxing Wu (*Lily*) and Hanchen Wu (*Cindy*) for their great support.

References

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

[2] A. Postlewaite and D. Schmeidler, Implementation in differential information economies. *Journal of Economic theory*, **39** (1986) 14-33.

[3] T.R. Palfrey and S. Srivastava, Implementation with incomplete information in exchange economies, *Econometrica*, **57** (1989) 115-134.

[4] M.O. Jackson, Bayesian implementation. *Econometrica*, **59** (1991) 461-477.

[5] 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

[6] H. Wu, On amending the Maskin's theorem by using complex numbers. *Games and Economic Behavior*, 2011 (submitted). http://arxiv.org/abs/1004.5327

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

[8] Mas-Colell, A., MD Whinston, and JR Green, Microeconomic Theory. Oxford University Press, Oxford, 1995.

[9] H. Matsushima, Bayesian monotonicity with side payments, *Journal of Economic Theory* **59** (1993) 107-121.

[10] 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. For example, suppose there are 3 agents. N={1, 2, 3}.
% Suppose the SCF f is incentive compatible, Bayesian monotonic and
%      multi-Bayesian monotonic. N^α={1, 2, 3}.  l = 3
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 agent 1's parameters. For example, ω̂_1 = Ĉ_3 = ω̂(0,π/3)
theta(1)=0;
phi(1)=pi/3;

% Reading agent 2's parameters. For example, ω̂_2 = Ĉ_3 = ω̂(0,π/3)
theta(2)=0;
phi(2)=pi/3;

% Reading agent 3's parameters. For example, ω̂_3 = Ĉ_3 = ω̂(0,π/3)
theta(3)=0;
phi(3)=pi/3;
```

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 (1))
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_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_n
```

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

```
% Computing |ψ2⟩ = [ω̂1 ⊗ ··· ⊗ ω̂n] Ĵ |ψ0⟩
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 $|\psi_2\rangle, |\psi_3\rangle, \langle\psi_3|\psi_3\rangle$.

```
% Randomly choosing a "collapsed" state according to the probability distribution ⟨ψ3|ψ3⟩
random_number=rand;
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 m1, ···, mn 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 $m_1, \cdots, m_n$.