MPRA
Munich Personal RePEc Archive

A classical algorithm to break through Maskin's theorem for small-scale cases

Wu, Haoyang

Department of Physics, Xi'an Jiaotong University, China

22 April 2010

Online at https://mpra.ub.uni-muenchen.de/22402/
MPRA Paper No. 22402, posted 22 Aug 2010 00:24 UTC

A classical algorithm to break through Maskin's theorem for small-scale cases

Haoyang Wu
Department of Physics, Xi'an Jiaotong University,
Xi'an, 710049, China.
hywch@mail.xjtu.edu.cn

Quantum mechanics has been applied to game theory for years. A recent work [H. Wu, Quantum mechanism helps agents combat "bad" social choice rules. *International Journal of Quantum Information*, 2010 (accepted). Also see http://arxiv.org/pdf/1002.4294v3] has generalized quantum mechanics to the theory of mechanism design (a reverse problem of game theory). Although the quantum mechanism is theoretically feasible, agents cannot benefit from it immediately due to the restriction of current experimental technologies. In this paper, a classical algorithm is proposed to help agents combat "bad" social choice rules immediately. The algorithm works well when the number of agents is not very large (e.g., less than 20). Since this condition is acceptable for small-scale cases, it can be concluded that the Maskin's sufficiency theorem has been broken through for small-scale cases just right now. In the future, when the experimental technologies for quantum information are commercially available, the Wu's quantum mechanism will break through the Maskin's sufficiency theorem completely.

Keywords: Quantum games; Prisoners' Dilemma; Mechanism design.

1. Introduction

About 2500 years ago, the Chinese philosopher Confucius proposed a logion to describe how atrociously a dictator dominated the civilians, i.e., "*tyranny was much fiercer than the tiger*". Fortunately, with the development of human societies, nowadays the dictatorship has almost been abandoned in the world. In a democratic society, a governor usually faces a "*social engineering*" problem, i.e., given some outcomes, whether he/she can design a mechanism that produces them.

It is an interesting question to ask whether a group of self-interested agents can find a way to fight the governor if all of them dislike a social choice rule (SCR) given by the governor. According to Maskin's sufficiency theorem [1], as long as an SCR is monotonic and satisfies no-veto, it is Nash implementable even if all agents dislike it (See Example 1 of Ref. [2]). Hence, the answer to the aforementioned question seems to be "No".

However, in 2010, Wu [2] generalized the theory of mechanism design to a quantum domain and proposed two results: 1) The success of Maskin's sufficiency theorem was indeed founded on a multi-player Prisoners' Dilemma, where self-interested agents could not enter a binding agreement to reach a Pareto-efficient outcome.

2) *By virtue of quantum strategies, agents who satisfied a certain condition could combat Pareto-inefficient SCRs instead of being restricted by Maskin's sufficiency theorem.* For n agents, the time and space complexity of the quantum mechanism was $O(n)$, therefore the quantum mechanism was theoretically feasible.

Despite these interesting results, there exists an obstacle for agents to use the quantum mechanism immediately: It needs a quantum equipment to work, but so far the experimental technologies for quantum information are not commercially available [5]. It is difficult for agents to carry out the quantum mechanism right now. As a result, the quantum mechanism may be viewed only as a "toy" to the real world.

In this paper, we will propose a classical algorithm through which agents can combat Pareto-inefficient SCRs immediately. Although the time and space complexity of the algorithm is exponential, it works well when the number of agents is not very large (e.g., less than 20). Since this condition is acceptable for small-scale cases, it can be concluded that the Maskin's sufficiency theorem has been broken through for small-scale cases just right now.

The rest of the paper is organized as follows: Section 2 recalls preliminaries of mechanism design; Section 3 is the main part of this paper, where we propose a classical algorithm to help agents combat "bad" SCRs when the number of agents is not very large; Section 4 draws conclusions.

2. Preliminaries

2.1. *The traditional mechanism design theory*

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 $g : \prod_{i\in N} M_i \to A$.

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

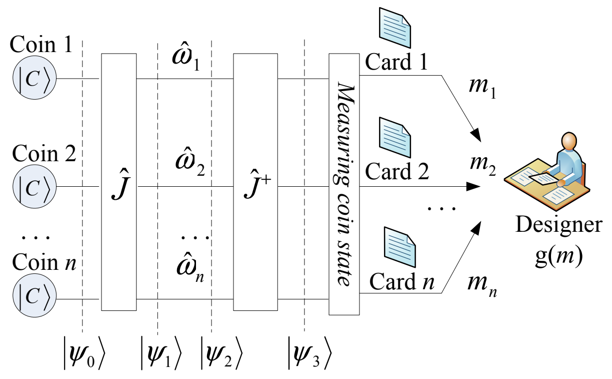

Fig. 1. The setup of a quantum mechanism. Each agent has a quantum coin and a card. Each agent independently performs a local unitary operation on his/her own quantum coin.

$g(\mathcal{N}(\Gamma, t))$ 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 mechanism as follows [3]:

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. *Wu's quantum mechanism*

According to Maskin's sufficiency theorem, even if all agents dislike an SCR specified by the designer, as long as it is monotonic and satisfies no-veto, the designer can always construct a mechanism to implement the SCR in Nash equilibrium. In 2010, Wu [2] proposed that when condition λ was satisfied, an original Nash implementable "bad" (i.e., Pareto-inefficient) SCR would no longer be Nash implementable in the context of a quantum domain. The setup of a quantum mechanism is depicted in Fig. 1. Assumptions of the quantum mechanism are referred to Ref. [2]. The working steps of the quantum mechanism are listed as follows:

Step 1: Nature selects a state $t \in \mathcal{T}$ and assigns t to the agents. The state of every quantum coin is set as $|C\rangle$. $|\psi_0\rangle = |C \cdots CC\rangle$.

Step 2: In state t, if all agents agree that the SCR F is "bad", i.e., there exists $\hat{t} \in \mathcal{T}$, $\hat{t} \neq t$, $\hat{a} \in F(\hat{t})$ such that $\hat{a} \succeq_i^t a \in F(t)$ for every $i \in N$, and $\hat{a} \succ_j^t a \in F(t)$ for at least one $j \in N$, then goto step 4.

Step 3: Each agent i sets $c_i = ((a_i, t_i, z_i), (a_i, t_i, z_i))$ (where $a_i \in A$, $t_i \in \mathcal{T}$, $z_i \in Z_+$), $\hat{\omega}_i = \hat{I}$, and sends $card(i, 0)$ as m_i to the designer. Goto step 8.

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 returns the collapsed state to the agents. Each agent i 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 $\hat{G}(\hat{s}) = g(m)$ using rule 1, 2 and 3. END.

3. Main results

3.1. *The matrix representations of quantum states*

In quantum mechanics, a quantum state can be described as a vector. For a two-qubit 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.

$$|C\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \hat{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \hat{\sigma}_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{1}$$

$$|\psi_0\rangle = \underbrace{|CC \cdots C\rangle}_{n} = \begin{bmatrix} 1 \\ 0 \\ \cdots \\ 0 \end{bmatrix}_{2^n \times 1} \tag{2}$$

$$\hat{J} = \cos(\gamma/2)\hat{I}^{\otimes n} + i\sin(\gamma/2)\hat{\sigma}_x^{\otimes n} \text{ (generalized from Ref. [4], Eq1)} \tag{3}$$

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

$$|\psi_1\rangle = \hat{J}\,\underbrace{|CC\cdots C\rangle}_{n} = \begin{bmatrix} \cos(\gamma/2) \\ 0 \\ \cdots \\ 0 \\ i\sin(\gamma/2) \end{bmatrix}_{2^n \times 1} \tag{5}$$

According to Ref. [4], the two-parameter quantum strategies of Eisert *et al.* are drawn from the set:

$$S^{(1)} = \{\hat{M}^{(1)}(\theta,\phi) : \theta \in [0,\pi], \phi \in [0,\pi/2]\}, \tag{6}$$

$$\hat{M}^{(1)}(\theta,\phi) = \begin{bmatrix} e^{i\phi}\cos(\theta/2) & i\sin(\theta/2) \\ i\sin(\theta/2) & e^{-i\phi}\cos(\theta/2) \end{bmatrix} \text{(Ref. [4], Eq4)} \tag{7}$$

Therefore,

$$\hat{\omega}_1 = \begin{bmatrix} e^{i\phi_1}\cos(\theta_1/2) & i\sin(\theta_1/2) \\ i\sin(\theta_1/2) & e^{-i\phi_1}\cos(\theta_1/2) \end{bmatrix}, \cdots, \hat{\omega}_n = \begin{bmatrix} e^{i\phi_n}\cos(\theta_n/2) & i\sin(\theta_n/2) \\ i\sin(\theta_n/2) & e^{-i\phi_n}\cos(\theta_n/2) \end{bmatrix}. \tag{8}$$

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 $|\psi_2\rangle = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}\,\underbrace{|CC\cdots C\rangle}_{n}$.

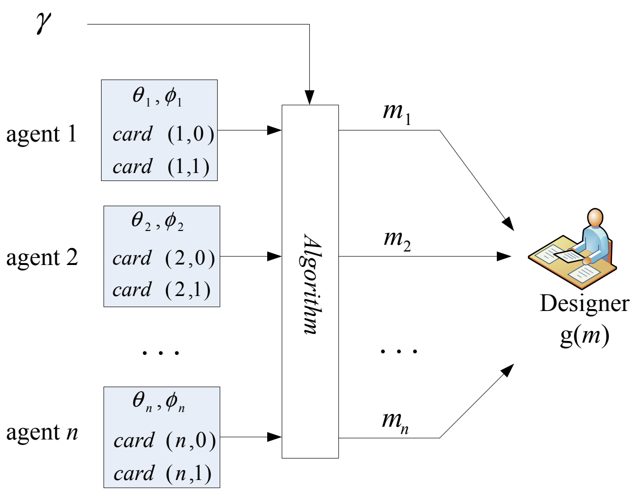

Fig. 2. The input and output of the algorithm.

$$\hat{J}^{+} = \begin{bmatrix} \cos(\gamma/2) & & & & & & & -i\sin(\gamma/2) \\ & \cos(\gamma/2) & & & & & -i\sin(\gamma/2) & \\ & & \cdots & & & \cdots & & \\ & & & \cos(\gamma/2) & -i\sin(\gamma/2) & & & \\ & & & -i\sin(\gamma/2) & \cos(\gamma/2) & & & \\ & & \cdots & & & \cdots & & \\ & -i\sin(\gamma/2) & & & & & \cos(\gamma/2) & \\ -i\sin(\gamma/2) & & & & & & & \cos(\gamma/2) \end{bmatrix}_{2^n \times 2^n} \quad (9)$$

$$|\psi_3\rangle = \hat{J}^{+}|\psi_2\rangle \quad (10)$$

3.2. *A classical algorithm*

Based on the matrix representations of quantum states, in the following we propose an algorithm that will generate the same outputs as the quantum mechanism does. The input and output of the algorithm are shown in Fig. 2. A *Matlab* program is given in Fig. 3(a)–(d).

Assumptions:

Each agent i ($i \in N$) has a card. 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 $c_i = (card(i, 0), card(i, 1))$, where $c_i \in A \times \mathcal{T} \times \mathbb{Z}_+ \times A \times \mathcal{T} \times \mathbb{Z}_+$, $card(i, 0) = (a_i, t_i, z_i)$, $card(i, 1) = (a'_i, t'_i, z'_i)$.

Inputs:

1) n: the number of agents;

2) γ: the coefficient of entanglement. (Note: Because the algorithm is irrelevant to experimental setups, the value of γ can be simply set as $\pi/2$.)

3) (θ_i, ϕ_i), $i = 1, \cdots, n$: the parameter of agent i's local operation $\hat{\omega}_i$.

Outputs:

m_i, $i = 1, \cdots, n$: the agent i's message.

Procedures:

Step 1: Nature selects a state $t \in \mathcal{T}$ and assigns t to the agents.

Step 2: In state t, if all agents agree that the SCR F is "bad", i.e., there exists $\hat{t} \in \mathcal{T}$, $\hat{t} \neq t$, $\hat{a} \in F(\hat{t})$ such that $\hat{a} \succeq_i^t a \in F(t)$ for every $i \in N$, and $\hat{a} \succ_j^t a \in F(t)$ for at least one $j \in N$, then goto step 4.

Step 3: Each agent i sends (a_i, t_i, z_i) (where $a_i \in A$, $t_i \in \mathcal{T}$, $z_i \in Z_+$) as m_i to the designer. Goto Step 11.

Step 4: Each agent i independently submits the parameter (θ_i, ϕ_i) of his/her local operation $\hat{\omega}_i$ to the algorithm (See Fig. 3(a)).

Step 5: Computing the matrix representation of $\hat{\omega}_1 \otimes \hat{\omega}_2 \otimes \cdots \otimes \hat{\omega}_n$. The full representation of this formula needs a $2^n \times 2^n$ matrix. However, since only the first and last value of $|\psi_1\rangle$ is non-zero, we can only compute the leftmost and rightmost columns of $\hat{\omega}_1 \otimes \hat{\omega}_2 \otimes \cdots \otimes \hat{\omega}_n$ (See Fig. 3(b)).

Step 6: Computing the vector representation of $|\psi_2\rangle = [\hat{\omega}_1 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}|C \cdots CC\rangle$.

Step 7: Computing the vector representation of $|\psi_3\rangle = \hat{J}^+|\psi_2\rangle$.

Step 8: Computing the probability distribution $\langle\psi_3|\psi_3\rangle$ (See Fig. 3(c)).

Step 9: The algorithm randomly chooses 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 10: 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)).

Step 11: The designer receives the overall message $m = (m_1, \cdots, m_n)$ and let the final outcome $\hat{G}(\hat{s}) = g(m)$ using rule 1, 2 and 3. END.

It can be seen from Step 11 that from the point of view of the designer, the interface between the designer and the agents is the same as its counterpart in the quantum mechanism. Therefore, the aforementioned classical algorithm will generate the same results as the quantum mechanism does. Although the time and space complexity of the algorithm is exponential, i.e., $O(2^n)$, when the number of agents is not very large (e.g., less than 20), the algorithm works well. For example, the runtime of the algorithm is about 0.5s for 15 agents, and about 12s for 20 agents (MATLAB 7.1, CPU: Intel (R) 2GHz, RAM: 3GB).

4. Conclusions

Confucius might be one of the earliest philosophers that felt pity for civilians subject to the dictatorship. Time elapsed, the Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 2007 was awarded jointly to Hurwicz, Maskin and

Myerson for having laid the foundations of mechanism design theory. However, the theory of mechanism design is somehow depressive for agents, because according to Maskin's sufficiency theorem, even if all agents dislike an SCR, as long as it is monotonic and satisfies no-veto, the designer can always construct a mechanism to implement it in Nash equilibrium (See Example 1 of Ref. [2]).

Although Wu's quantum mechanism is interesting and theoretically feasible, current experimental technologies restrict it to be practically available for agents. In this paper, we go beyond the obstacle of how to realize the quantum mechanism, and propose an algorithm through which agents can combat "bad" SCRs in a classical computer if the number of agents is not very large. As a result, people do not have to construct a real quantum equipment to benefit from the quantum mechanism. The novel algorithm can be easily applied to small-scale applications just right now. In the future, when the experimental technologies of quantum information are available for large-scale cases, Maskin's sufficiency theorem will be completely broken through by using the quantum mechanism.

References

1. E. Maskin, *Rev. Econom. Stud.* **66** (1999) 23-38.
2. H. Wu, Quantum mechanism helps agents combat "bad" social choice rules. *International Journal of Quantum Information*, 2010 (accepted). Also see http://arxiv.org/pdf/1002.4294v3
3. R. Serrano, *SIAM Review* **46** (2004) 377-414.
4. A.P. Flitney and L.C.L. Hollenberg, *Phys. Lett. A* **363** (2007) 381-388.
5. T.D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe and J.L. O'Brien, *Nature*, **464** (2010) 45-53.

```
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 pi/2.
gamma=pi/2;

% Defining the array of $(\theta_i, \phi_i), i = 1, \cdots, n$
theta=zeros(n,1);
phi=zeros(n,1);

% Apple independently submits her parameters of local operation. For example, $\hat{\omega}_1 = \hat{C}_2 = \hat{\omega}(0, \pi/2)$
theta(1)=0;
phi(1)=pi/2;

% Lily independently submits her parameters of local operation. For example, $\hat{\omega}_2 = \hat{C}_2 = \hat{\omega}(0, \pi/2)$
theta(2)=0;
phi(2)=pi/2;

% Cindy independently submits her parameters of local operation. For example, $\hat{\omega}_3 = \hat{I} = \hat{\omega}(0,0)$
theta(3)=0;
phi(3)=0;
```

Fig. 3 (a). Initialization of the algorithm.

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

% In the beginning, A represents the local operation $\hat{\omega}_1$ of agent 1. (See Eq 8)
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 $\hat{\omega}_1 \otimes \hat{\omega}_2 \otimes \cdots \otimes \hat{\omega}_n$
for agent=2 : n
        % B varies from $\hat{\omega}_2$ to $\hat{\omega}_n$
        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 $\hat{\omega}_1 \otimes \hat{\omega}_2 \otimes \cdots \otimes \hat{\omega}_n$
```

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

```
% Computing $|\psi_2\rangle = [\hat{\omega}_1 \otimes \hat{\omega}_2 \otimes \cdots \otimes \hat{\omega}_n]\hat{J}|C \cdots CC\rangle$
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 $|\psi_3\rangle = \hat{J}^+|\psi_2\rangle$
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 $\langle\psi_3|\psi_3\rangle$
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 $\langle\psi_3|\psi_3\rangle$
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 $|C\rangle$, '1' stands for $|D\rangle$
indexstr=dec2bin(index-1);
sizeofindexstr=size(indexstr);

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

% For each agent $i \in N$, 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 $|C\rangle$
      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 outputs all messages $m_1, m_2, \cdots, m_n$ 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 $m_1, m_2, \cdots, m_n$. This part corresponds to Step 7 of Wu's quantum mechanism.