Towards Truthful Mechanisms for Binary Demand Games: A General Framework

Ming-Yang Kao*
Dept. of Computer Science
Northwestern University
Evanston, IL, USA
kao@cs.northwestern.edu

Xiang-Yang Li†
Dept. of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
xli@cs.iit.edu

WeiZhao Wang
Dept. of Computer Science
Illinois Institute of Technology
Chicago, IL, USA
wangwei4@iit.edu

ABSTRACT

The family of Vickrey-Clarke-Groves (VCG) mechanisms is arguably the most celebrated achievement in truthful mechanism design. However, VCG mechanisms have their limitations. They only apply to optimization problems with a utilitarian objective function, and their output should optimize the objective function. For many optimization problems, finding the optimal output is computationally intractable. If we apply VCG mechanisms to polynomial-time algorithms that approximate the optimal solution, the resulting mechanisms may no longer be truthful.

In light of these limitations, it is useful to study whether we can design a truthful non-VCG payment scheme that is computationally tractable for a given output method $\mathcal{O}$. In this paper, we focus our attention on *binary demand games* in which the agents' only available actions are to take part in the a game or not to. For these problems, we prove that a truthful mechanism $M = (\mathcal{O}, \mathcal{P})$ exists (with proper payment method $\mathcal{P}$) if and only if $\mathcal{O}$ satisfies a certain monotone property. We also provide several general algorithms to compute the payments efficiently for various types of output. In particular, we show how a truthful payment can be computed through "or/and" combinations, round-based combinations, and some more complex combinations of outputs from subgames.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]: General; J.4 [**Social and Behavioral Sciences**]: Economics; K.4.4 [**Computer and Society**]: Electronic Commerce

General Terms

Algorithms, Economics

Keywords

Selfish agent, mechanism design, pricing, demand games, general framework.

*Supported in part by NSF Grant IIS-0121491.

†Supported in part by NSF under Grant CCR-0311174.

EC'05, June 5–8, 2005, Vancouver, British Columbia, Canada.

1. INTRODUCTION

In recent years, with the rapid development of the Internet, many protocols and algorithms have been proposed to make this Internet working more efficient and secure. The Internet is a complex distributed system where a multitude of heterogeneous agents cooperate together to achieve some common goals, and these protocols and algorithms often assume that all agents will follow the prescribed rules without any deviation. However, in some settings where the agents are selfish instead of altruistic, it is more reasonable to assume these agents are *rational* – maximize their own profits – according to the neoclassic economics, and new models are needed to cope with selfish behavior of these agents.

Towards this end, Nisan and Ronen [13] proposed the framework of *algorithm mechanism design* and applied VCG mechanism to some fundamental problems in computer science, including shortest paths, minimum spanning trees, and scheduling on unrelated machines. The VCG mechanisms [17, 5, 10] apply to mechanism design problems whose outputs optimize the objective function $g(o, t)$, which is simply the sum of all agents' valuations and is known as *utilitarian*. Unfortunately, some objective functions are not utilitarian; even for those problems with utilitarian objective function, sometimes it is impossible to find the optimal output in polynomial time unless P=NP. Thus, some methods other than VCG mechanism are needed to address these issues.

Archer and Tardos [2] studied a scheduling problem where it is NP-Hard to find the optimal output. They pointed out that a certain monotone property of the output work load w_i is a necessary and sufficient condition for the existence of a truthful mechanism for their scheduling problem. Auletta *et al.* [3] studied a similar problem. They provided a family of deterministic truthful $(2 + \epsilon)$-approximation mechanisms for any fixed number of machines and several $(1 + \epsilon)$-truthful mechanisms for some NP-hard restrictions of their scheduling problem. Lehmann *et al.*[11] studied the single-minded combinatorial auction and gave a $\sqrt{m}$-approximation truthful mechanism. They also pointed out that a certain monotonicity in the allocation in single-minded combinatorial auction can lead to a truthful mechanism. The work of Mu'alem and Nisan [12] is the closest in spirit to our work. They characterized all truthful mechanisms based on a certain monotone property in a single-minded auction setting. They also showed how to used MAX and IF-THEN-ELSE to combine outputs from subproblems. As we will show later, the MAX and IF-THEN-ELSE combinations could be treated as a special case of the combination schemes we presents in this paper. More generally, our main contributions in this paper are several algorithms for computing the payment in polynomial time under mild assumptions.

In our paper, we study how to design strategyproof mechanisms for *binary demand games* where the output of an agent is either

"selected" or "not selected". We also assume that the valuations of agents are uncorrelated, i.e., the valuation of an agent only depends on its own output and type. Recall that a mechanism $M = (\mathcal{O}, \mathcal{P})$ is composed of two parts, an output function $\mathcal{O}$ and a payment scheme $\mathcal{P}$. Previously, it is often assumed that there is an objective function g and an output $\mathcal{O}$, that either optimizes or is an α-approximation of g. In sharp contrast to VCG mechanisms, we do not require that the output should optimize the objective function. In fact, we do not even require the existence of an objective function. Given any output method $\mathcal{O}$ for a binary demand game, we showed that a truthful mechanism $M = (\mathcal{O}, \mathcal{P})$ exists for the game if and only if $\mathcal{O}$ satisfies a certain *monotone property*. Note that the monotone property only guarantees the existence of a payment $\mathcal{P}$ that is strategyproof, it does not give any method to compute $\mathcal{P}$. Even with the knowledge of the existence of a truthful mechanism, sometimes it could still be very hard, if not impossible, to find the payment $\mathcal{P}$. Thus, instead of giving a method to design and compute the truthful mechanisms for all binary demand games, we give a general framework to design the truthful mechanisms. In particular, we present algorithms to compute the payment when the output is a composition of the outputs of subgames through the operators "or" and "and"; through round-based combinations; through intermediate results, which may be again computed from other subproblems.

The remainder of the paper is organized as follows. In Section 2, we discuss preliminaries and previous works, define binary demand games and discuss the basic assumptions about binary demand games. In Section 3, we first study some basic properties of the truthful mechanisms $M = (\mathcal{O}, \mathcal{P})$ for a binary demand game when the output function $\mathcal{O}$ is given. We then show that $\mathcal{O}$ satisfying a certain monotone property is a necessary and sufficient condition for the existence of a truthful mechanism $M = (\mathcal{O}, \mathcal{P})$. A framework is then proposed to compute the payment $\mathcal{P}$ in polynomial time for several types of output methods $\mathcal{O}$. In Section 4, we provide several examples to demonstrate the effectiveness of our general framework. We conclude our paper in Section 5 with some possible future directions.

2. PRELIMINARIES

2.1 Mechanism Design

In the design of centralized or distributed network protocols with inputs from individual agents, the computational agents are typically assumed to be either *correct/obedient* or *faulty* (also called adversarial). In contrast to this conventional approach, this paper follows the assumption in neoclassic economics that all agents are *rational*, i.e., they respond to well-defined incentives and will deviate from the protocol if and only if it improves their gain.

A standard model for mechanism design is as follows. Assume there are n agents $\{1, \cdots, n\}$ and every agent i has some private information t_i, called its *type*, only known to itself. For example, the type t_i can be the cost agent i forwarding a packet in a network or can be a payment that the agent is willing to pay for a good in an auction. The set of all agents' types defines the *type vector* $t = (t_1, t_2, \cdots, t_n)$. Every agent i has a set of strategies A_i he can choose from. For each input vector $a = (a_1, \cdots, a_n)$ where agent i plays strategy $a_i \in A_i$, the mechanism $\mathcal{M} = (\mathcal{O}, \mathcal{P})$ computes an *output* $o = \mathcal{O}(a)$ and a *payment* vector $p = (p_1, \cdots, p_n)$, where $p_i = \mathcal{P}_i(a_1, \cdots, a_n)$. Here the payment p_i is the money given to participating agent i and depends on the strategies used by agents. If $p_i < 0$, the agent has to pay $-p_i$ to participate in the action. A game is defined as $\mathcal{G} = (\mathcal{S}, \mathcal{M})$, where $\mathcal{S}$ is setting for game $\mathcal{G}$. Here, $\mathcal{S}$ are the parameters of the game that are settled down before the game starts and do not depend on the players' strategies. For example, in a unicast routing game [13], the setting includes the topology of the network, the source node and the destination node. In summary, for a game $\mathcal{G}$, given the strategy vector a of the players, it computes the output and payment for every player. Throughout this paper, if we dot not mention explicitly, the setting $\mathcal{S}$ of the game is always fixed and we are only interested in how to design the mechanism.

For each possible output o, agent i's preferences are given by a valuation function $v(t_i, o)$ that assigns a real monetary number to output o. Everything in the scenario, including the setting $\mathcal{S}$, the output method $\mathcal{O}$ and payment method $\mathcal{P}$, is public knowledge except the type t_i, which is the private information to agent i. Let $u_i(t_i, o)$ denote the *utility* of agent i at the outcome of the game o, given its preferences t_i. Hereafter. we assume the utility for agent i is $u_i(t_i, o) = v(t_i, o) + p_i(a)$ (i.e., quasi-linear).

Let $a|^i a'_i = (a_1, \cdots, a_{i-1}, a'_i, a_{i+1}, \cdots, a_n)$, i.e., each agent $j \neq i$ plays strategy a_j except that the agent i plays a'_i. Let $a_{-i} = (a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_n)$ denote the strategies of all other agents except i. Sometimes, we write (a_i, b_{-i}) as $b|^i a_i$. A strategy a_i is called a *dominant strategy* if it (weakly) maximizes the utility of i for all possible strategies of other agents, i.e., $u_i(t_i, \mathcal{O}(a_i, b_{-i})) \geq u_i(t_i, \mathcal{O}(a'_i, b_{-i}))$ for all $a'_i \neq a_i$ and all strategies b_{-i} of agents other than i. A strategy vector a^* is called a *Nash equilibrium* if it (weakly) maximizes the utility when the strategies of all agents are fixed as a^*_{-i}, i.e., $u_i(t_i, \mathcal{O}(a^*)) \geq u_i(t_i, \mathcal{O}(a'_i, a^*_{-i}))$ for all i, and $a'_i \neq a^*_i$.

A direct-revelation mechanism is a mechanism in which the only actions available to the agents are to make direct claims about its private type t_i to the mechanism. A direct-revelation mechanism is *incentive compatible* (IC) if each agent maximizes its utility when it reports its type t_i to the mechanism truthfully. A direct-revelation mechanism is strategy-proof if truth-revelation is a dominant-strategy equilibrium. Then, in a direct-revelation strategy-proof mechanism, the payment function should satisfy the property that, for each agent i, $v(t_i, \mathcal{O}(t)) + p_i(t) \geq v(t_i, \mathcal{O}(t|^i t'_i)) + p_i(t|^i t'_i)$. Another common requirement in the literature for mechanism design is so called *individual rationality* or *voluntary participation*: the agent's utility of participating in the output of the mechanism is not less than the utility of the agent if it did not participate.

Arguably the most important positive result in mechanism design is the generalized Vickrey-Clarke-Groves (VCG) mechanism by Vickrey [17], Clarke [5], and Groves [10]. The VCG mechanism applies to maximization problems where the objective function is *utilitarian* $g(o, t) = \sum_i v(t_i, o)$ (i.e., the sum of all agents' valuations) and the set of possible outputs is assumed to be finite. A direct revelation mechanism $\mathcal{M} = (\mathcal{O}(t), \mathcal{P}(t))$ belongs to the VCG family if (1) the output $\mathcal{O}(t)$ maximizes $\sum_i v(t_i, o)$, and (2) the payment to agent i is $p_i(t) = \sum_{j \neq i} v_j(t_j, \mathcal{O}(t)) + h^i(t_{-i})$, where $h^i()$ is an arbitrary function of t_{-i}. Under mild assumptions, VCG mechanisms are the *only* truthful implementations for utilitarian problems [9].

An output function of a VCG mechanism is required to maximize the objective function. This makes the mechanism computationally intractable in many cases. Notice that replacing the optimal algorithm with non-optimal approximation usually leads to untruthful mechanisms if VCG payment method is used. In their seminal paper on algorithmic mechanism design, Nisan and Ronen [13] add computational efficiency to the set of concerns that must be addressed in the study of how privately known preferences of a large group of selfish agents can be aggregated into a "social choice" that results in optimal allocation of resources. In this paper, we study how to design a strategy-proof mechanism that does *not*

necessarily implement the utilitarian objective function.

2.2 Previous Work

Lehmann *et al.* [11] studied how to design an efficient truthful mechanism for single-minded combinatorial auction. In a single-minded combinatorial auction, each agent i ($1 \leq i \leq n$) only wants to buy a subset $S_i \subseteq S$ with private price c_i. A single-minded bidder i declares a bid $b_i = \langle S_i', a_i \rangle$ with $S_i' \subseteq S$ and $a_i \in R^+$. In [11], it is assumed that the set of goods allocated to an agent i is either S_i' or $\emptyset$, which is known as *exactness*. Lehmann *et al.* gave a greedy round-based allocation algorithm, based on the rank $\frac{a_i}{|S_i'|^{1/2}}$, that has an approximation ratio $\sqrt{m}$, where m is the number of goods in S. Based on the approximation algorithm, they gave a truthful payment scheme. For an allocation rule satisfying (1) *exactness*: the set of goods allocated to an agent i is either S_i' or $\emptyset$; (2) *monotonicity*: proposing more money for fewer goods cannot cause a bidder to lose its bid, they proposed a truthful payment scheme as follows: (1) charge a winning bidder a certain amount that does not depend on its own bidding; (2) charge a losing bidder 0. Notice the assumption of *exactness* reveals that the single minded auction is indeed a binary demand game. Their payment scheme inspired our payment scheme for binary demand game.

In [1], Archer *et al.* studied the combinatorial auctions where multiple copies of many different items are on sale, and each bidder i desires only one subset S_i. They devised a randomized rounding method that is incentive compatible and gave a truthful mechanism for combinatorial auctions with single parameter agents that approximately maximizes the social value of the auction. As they pointed out, their method is *strongly truthful* in sense that it is truthful with high probability $1 - \epsilon$, where ϵ is an error probability. On the contrary, in this paper, we study how to design a *deterministic* mechanism that is truthful based on some given outputs.

In [2], Archer and Tardos showed how to design truthful mechanisms for several combinatorial problems where each agent's private information is naturally expressed by *a single positive real number*, which will always be the cost incurred per unit load. The mechanism's output could be arbitrary real number but their valuation is a quisilinear function $t \cdot w$, where t is the private per unit cost and w is the work load. Archer and Tardos characterized that all truthful mechanism should have decreasing "work curves" w and that the truthful payment should be $P_i(b_i) = P_i(0) + b_i w_i(b_i) - \int_0^{b_i} w_i(u)du$ Using this model, Archer and Tardos designed truthful mechanisms for several scheduling related problems, including minimizing the span, maximizing flow and minimizing the weighted sum of completion time problems. Notice when the load of the problems is $w = \{0, 1\}$, it is indeed a binary demand game. If we apply their characterization of the truthful mechanism, their decreasing "work curves" w implies exactly the monotone property of the output. But notice that their proof is heavily based on the assumption that the output is a continuous function of the cost, thus their conclusion can't directly apply to binary demand games.

The paper of Ahuva Mu'alem and Noam Nisan [12] is closest in spirit to our work. They clearly stated that *we only discussed a limited class of bidders, single minded bidders, that was introduced by* [11]. They proved that all truthful mechanisms should have a monotone output and their payment scheme is based on the cut value. With a simple generalization, we get our conclusion for general binary demand game. They proposed several combination methods including MAX, IF-THEN-ELSE construction to perform partial search. All of their methods required the welfare function associated with the output satisfying *bitonic* property.

2.3 Binary Demand Games

Given the strategies of all agents for a game $\mathcal{G}$, a mechanism computes the output and the payment. Generally, the output function $\mathcal{O}$ maps each given type vector t to an output from a given output space. In this paper, we focus our attention on the set of output functions, whose range is $\{0, 1\}^n$. In other words, the output is a n-tuple vector $\mathcal{O}(t) = (\mathcal{O}_1(t), \mathcal{O}_2(t), \cdots, \mathcal{O}_n(t))$, where $\mathcal{O}_i(t) = 1$ (resp. 0) means that agent i is(resp. not) selected. We call such a mechanism design problem a *binary demand game*. Examples of binary demand games include: unicast [13, 18, 8] and multicast [19, 20, 7] (generally subgraph construction by selecting some links/nodes to satisfy some property), facility location [6], and a certain auction [11, 2, 12].

Hereafter, we also assume that the valuation of each agent i for $o_i = 0$ is a publicly known value. This assumption is needed to design a mechanism satisfying the individual rationality (IR) property. If this assumption does not hold, in many applications, the agent can get arbitrarily large utility by declaring arbitrarily small $v(t_i, 0)$ and $v(t_i, 1)$ when the valuation of agents are negative. For example, in the unicast problem, the IR property implies that the payment to an agent is at least $\min(-v(t_i, 0), -v(t_i, 1))$. Without loss of generality, we assume that the valuation $v(t_i, 0)$ is normalized to 0. Thus, throughout his paper, we only consider these direct-revelation mechanisms in which every agent only needs to reveal its valuation $v_i = v(t_i, 1)$ when it is selected. Notice that in applications where agents providing service and receiving payment, e.g., unicast and job scheduling, the valuation v_i of an agent i is usually negative. For the convenience of our presentation, we introduce the cost of agent which is $c_i = -v(t_i, 1)$, i.e., it costs agent i c_i to provide the service. Throughout this paper, we will use c_i instead of v_i in our analysis. All our results can apply to the case when the agents procure the service instead of provide by setting c_i to negative, as in auction.

A binary demand game is called an *binary optimization demand game* if we optimize an object function. The main differences between binary demand games and those problems that can be solved by VCG mechanisms are:

1. The objective function is *utilitarian* for a problem solvable by VCG while there is no restriction on the objective function for a binary demand game.
2. The output function $\mathcal{O}$ does not necessarily optimize an objective function, while a VCG mechanism only uses the output that optimizes the objective function.
3. The range of the output method in a binary demand game is $\{0, 1\}^n$, while the range of the output function in a VCG mechanism may be arbitrary.
4. We assume that the agents' valuations are not correlated in a binary demand game, while the agents' valuations may be correlated in a VCG mechanism.

In this paper, we assume for technical convenience that the objective function $g(o, c)$, if exists, is continuous regarding the cost c_i, but most of our results can be directly applied to the discrete case without any modifications.

3. GENERAL APPROACHES

3.1 Properties of Strategyproof Mechanisms

Generally, there are several properties that direct revelation mechanisms need to satisfy in order to be truthful (see appendix for the proofs of the following theorems).

THEOREM 1. *If a mechanism $M = (\mathcal{O}, \mathcal{P})$ satisfies IC, then $\forall i$, if $\mathcal{O}_i(t|^i t_{i_1}) = \mathcal{O}_i(t|^i t_{i_2})$, then $p_i(t|^i t_{i_1}) = p_i(t|^i t_{i_2})$.*

COROLLARY 2. *For any strategy-proof mechanism for a binary demand game $\mathcal{G}$ with setting $\mathcal{S}$, if we fix the cost c_{-i} of all agents other than i, the payment to agent i is a constant p_i^1 if $\mathcal{O}_i(c) = 1$, and it is another constant p_i^0 if $\mathcal{O}_i(c) = 0$.*

THEOREM 3. *Fixed the setting $\mathcal{S}$ for a binary demand game, if mechanism $M = (\mathcal{O}, \mathcal{P})$ satisfies IC, then mechanism $M' = (\mathcal{O}, \mathcal{P}')$ with the same output method $\mathcal{O}$ and $p'_i(c) = p_i(c) - \delta_i(c_{-i})$ for any function $\delta_i(c_{-i})$ also satisfies IC.*

The proof of this theorem is straightforward and thus omitted. This theorem implies that for the binary demand games we can always normalize the payment to an agent i such that the payment to the agent is 0 when it is not selected. Hereafter, we will only consider normalized payment schemes, i.e., $p_i^0 = 0$.

3.2 Existence of Strategyproof Mechanisms

Notice, given the setting $\mathcal{S}$, a mechanism design problem is composed of two parts: the output function $\mathcal{O}$ and a payment function $\mathcal{P}$. In this paper, we use a given output $\mathcal{O}$ and focus our attention on how to design a truthful payment scheme based on $\mathcal{O}$. Given an output method $\mathcal{O}$ for a binary demand game, we first present a sufficient and necessary condition for the existence of a truthful payment scheme $\mathcal{P}$.

DEFINITION 1 (MONOTONE NON-INCREASING PROPERTY (MP)). *An output method $\mathcal{O}$ is said to satisfy the* monotone non-increasing property *if for every agent i and two of its possible costs $c_{i_1} < c_{i_2}$, $\mathcal{O}_i(c|^i c_{i_2}) \leq \mathcal{O}_i(c|^i c_{i_1})$.*

This definition is not restricted only to binary demand games. For binary demand games, this definition implies that if $\mathcal{O}_i(c|^i c_{i_2}) = 1$ then $\mathcal{O}_i(c|^i c_{i_1}) = 1$.

THEOREM 4. *For any agent i and any fixed c_{-i} in a binary demand game with output method $\mathcal{O}$, the following three conditions are equivalent:*

1. *There exists a value $\kappa_i(\mathcal{O}, c_{-i})$ (called a* cut value*) such that $\mathcal{O}_i(c) = 1$ if $c_i < \kappa_i(\mathcal{O}, c_{-i})$ and $\mathcal{O}_i(c) = 0$ if $c_i > \kappa_i(\mathcal{O}, c_{-i})$. When $c_i = \kappa_i(\mathcal{O}, c_{-i})$, $\mathcal{O}_i(c)$ could be either 0 or 1, depending on the tie-breaker of the output method $\mathcal{O}$. Hereafter, we will not consider the tie-breaker scenario in our proofs.*
2. *The output method $\mathcal{O}$ satisfies MP.*
3. *There exists a truthful payment scheme $\mathcal{P}$.*

See the appendix for the proof of the theorem. We now summarize the process to design a truthful payment scheme for a binary demand game based on an output method $\mathcal{O}$.

Regarding our general framework, we have the following theorem.

THEOREM 5. *The payment defined Algorithm 1 is minimum among all truthful payment schemes using $\mathcal{O}$ as output.*

Here, we briefly illustrate our general framework through the example of unicast routing, which is well studied [13]. We assume that the nodes are the agents, but all arguments and algorithms can be applied to the case when the links are the agents. Assume that, the output of the mechanism uses the path connecting the source and target with the minimum cost, called the *least cost path* (LCP) hereafter.

Algorithm 1 GENERAL FRAMEWORK: Truthful mechanism design for a binary demand game

1: Check whether the output method $\mathcal{O}$ satisfies the monotone property. If it does not, then there is no payment scheme $\mathcal{P}$ such that mechanism $M = (\mathcal{O}, \mathcal{P})$ is truthful. Otherwise, define the payment $\mathcal{P}$ as follows.
2: Based on the method $\mathcal{O}$, find the cut value $\kappa_i(\mathcal{O}, c_{-i})$ for agent i such that $\mathcal{O}_i(c|^i d_i) = 1$ when $d_i < \kappa_i(\mathcal{O}, c_{-i})$, and $\mathcal{O}_i(c|^i d_i) = 0$ when $d_i > \kappa_i(\mathcal{O}, c_{-i})$.
3: The payment for agent i is 0 if $\mathcal{O}_i(c) = 0$; the payment is $\kappa_i(\mathcal{O}, c_{-i})$ if $\mathcal{O}_i(c) = 1$.

It is easy to show that LCP satisfies the monotone property. Thus, there exists a payment scheme $\mathcal{P}$ such that $M = (LCP, \mathcal{P})$ is a truthful mechanism. Now, we find the cut value when LCP is used as the output. Let $\mathsf{LCP}(q_i, q_j, c|^k 0)$ be the least cost path when q_k is selected with cost 0. Let $\mathsf{LCP}_{-q_k}(q_i, q_j, c)$ be the least cost path without node q_k. Obviously, when the cost of node q_k is no more than $|\mathsf{LCP}_{-q_k}(q_i, q_j, c)| - |\mathsf{LCP}(q_i, q_j, c|^k 0)|$, node q_k is selected. Consequently, $\kappa_k(LCP, c_{-k}) = |\mathsf{LCP}_{-q_k}(q_i, q_j, c)| - |\mathsf{LCP}(q_i, q_j, c|^k 0)|$. The payment for q_k is $p_k = \kappa(LCP, c_{-k})$. Notice $|\mathsf{LCP}(q_i, q_j, c|^k 0)| = |\mathsf{LCP}(q_i, q_j, c)| - c_k$ when q_k is selected under cost profile c. Thus, the payment p_k can be written as, when q_k is selected, $|\mathsf{LCP}_{-q_k}(q_i, q_j, c)| - |\mathsf{LCP}(q_i, q_j, c)| + c_k$, which is exactly the same as what we get from the VCG mechanism.

3.3 Computing Cut Value Functions

It seems quite easy to find the truthful payment scheme by using our general framework, but as we will see later, the most difficult part is how to find the cut value function when we know it does exist. Notice that the simple binary search methods do not work generally since the valuation of the agents could be continuous. Instead of trying to give a universal approach to compute the cut value for all output methods satisfying MP, we give some useful general techniques that can help us finding cut value under certain circumstances. Our basic approach is as follows. First, we decompose the output methods into the composition of several output methods. Then we find the cut value function for each of the decomposed output methods. Finally, we calculate the original cut value function by combining the cut value functions of these decomposed functions.

3.3.1 Simple Combinations

Given an output function $\mathcal{O}$, let $\kappa(\mathcal{O}, c)$ denote a n-tuple vector $(\kappa_1(\mathcal{O}, c_{-1}), \kappa_2(\mathcal{O}, c_{-2}), \cdots, \kappa_n(\mathcal{O}, c_{-n}))$. Here, $\kappa_i(\mathcal{O}, c_{-i})$ is the cut value for agent i when the output function is $\mathcal{O}$ and the costs c_{-i} of all other agents are fixed. Regarding the combination of binary demand game, we have the following theorem.

THEOREM 6. *Fixed the setting $\mathcal{S}$ of a binary demand game, assume that there are m output functions $\mathcal{O}^1, \mathcal{O}^2, \cdots, \mathcal{O}^m$ satisfying the monotone property, and $\kappa(\mathcal{O}^i, c)$ is the cut value function vector for $\mathcal{O}^i$ where $i = 1, 2, \cdots, m$. Then the output function $\mathcal{O}(c) = \bigvee_{i=1}^m \mathcal{O}^i(c)$ also satisfies the monotone property. Moreover, the cut value function for $\mathcal{O}$ is*

$$\kappa(\mathcal{O}, c) = \max_{i=1}^{m} \{\kappa(\mathcal{O}^i, c)\}.$$

Here $\kappa(\mathcal{O}, c) = \max_{i=1}^{m}\{\kappa(\mathcal{O}^i, c)\}$ *means,* $\forall j \in [1, n]$,

$$\kappa_j(\mathcal{O}, c_{-j}) = \max_{i=1}^{m}\{\kappa_j(\mathcal{O}^i, c_{-j})\}$$

and $\mathcal{O}(c) = \bigvee_{i=1}^{m} \mathcal{O}^i(c)$ *means,* $\forall j \in [1, n]$,

$$\mathcal{O}_j(c) = \mathcal{O}_j^1(c) \vee \mathcal{O}_j^2(c) \vee \cdots \vee \mathcal{O}_j^m(c).$$

The proof of this theorem is omitted due to space limit. In practice, many algorithms indeed fall into this category. To show the usefulness of Theorem 6, we study a concrete example here. In a network, sometimes we want to deliver the packet to a set of terminals instead of one, which is known as multicast. The most commonly used structure in multicast routing is so called Shortest Path Tree (SPT). Consider a network $G = (V, E, c)$, where V is the set of terminals, and c is the actual cost of the node forwarding the data. Assume that the source node is s and the receivers are $Q \subset V$. For each receiver $q_i \in Q$, we compute the shortest path (least cost path), denoted by $\mathsf{LCP}(s, q_i, d)$, from the source s to q_i under the reported cost profile d. The union of all such shortest paths forms the shortest path tree. We then use our general approach to design the truthful payment scheme $\mathcal{P}$ when the SPT structure is used as the output for multicast, i.e., a mechanism $M = (SPT, \mathcal{P})$.

We define $LCP^{(s,q_i)}$ as the output generated by $\mathsf{LCP}(s, q_i, d)$, i.e., $LCP_k^{(s,q_i)}(d) = 1$ if and only if node v_k is in $\mathsf{LCP}(s, q_i, d)$. Then the output SPT is defined as $\bigvee_{q_i \in Q} LCP^{(s,q_i)}$. In other words, $SPT_k(d) = 1$ if and only if q_k is selected in some $\mathsf{LCP}(s, q_i, d)$. In previous section, we have already proven that shortest path satisfies MP. Thus, by applying Theorem 6, we know that SPT also satisfies MP, and the cut value function vector for SPT can be calculated as $\kappa(SPT, c) = \max_{q_i \in Q} \kappa(LCP^{(s,q_i)}, c)$, where $\kappa(LCP^{(s,q_i)}, c)$ is the cut value function vector for the shortest path $\mathsf{LCP}(s, q_i, c)$. Consequently, the payment scheme above is truthful and minimal among all truthful payment scheme when the output is SPT.

THEOREM 7. *Fixed the setting $\mathcal{S}$ of a binary demand game, assume that there are m output methods $\mathcal{O}^1, \mathcal{O}^2, \cdots, \mathcal{O}^m$ satisfying MP, and $\kappa(\mathcal{O}^i, c)$ are the cut value functions respectively for $\mathcal{O}^i$ where $i = 1, 2, \cdots, m$. Then the output function $\mathcal{O}(c) = \bigwedge_{i=1}^{m} \mathcal{O}^i(c)$ satisfies MP. Moreover, the cut value function for $\mathcal{O}$ is $\kappa(\mathcal{O}, c) = \min_{i=1}^{m}\{\kappa(\mathcal{O}^i, c)\}$.*

We then show that our simple combination generalizes the IF-THEN-ELSE function defined in [12]. For an agent i, assume that there are two output functions $\mathcal{O}^1$ and $\mathcal{O}^2$ satisfying MP. Let $\kappa_i(\mathcal{O}^1, c_{-i})$, $\kappa_i(\mathcal{O}^2, c_{-i})$ be the cut value functions for $\mathcal{O}^1$, $\mathcal{O}^2$ respectively. Then the IF-THEN-ELSE function $\mathcal{O}_i(c)$ could be treated as $\mathcal{O}_i(c) = [(c_i \leq \kappa_i(\mathcal{O}^1, c_{-i}) + \delta_1(c_{-i})) \wedge \mathcal{O}^2(c_{-i}, c_i)] \vee (c_i < \kappa_i(\mathcal{O}^1, c_{-i}) - \delta_2(c_{-i}))$ where $\delta_1(c_{-i})$ and $\delta_2(c_{-i})$ are two positive functions. By applying Theorems 6 and 7, we know that the output function $\mathcal{O}$ satisfies MP and $\kappa_i(\mathcal{O}, c_{-i})$ is

$$\max\{\min(\kappa_i(\mathcal{O}^1, c_{-i}) + \delta_1(c_{-i}), \kappa_i(\mathcal{O}^2, c_{-i})), \kappa_i(\mathcal{O}^1, c_{-i}) - \delta_2(c_{-i}))\}.$$

3.3.2 Cut-Value for Round-Based Outputs

The majority approximation algorithms can be categorized as round-based approaches: in each round the algorithm selects some agents, and then updates the setting and the cost profile if necessary. For example, majority approximation algorithms for minimum weighted vertex cover, maximum weighted independent set, minimum weighted dominating set, minimum weighted set cover, minimum weighted Steiner tree, and so on fall into this category.

We study by an example to show how to compute the cut value for a round-based output. The example we used here is the *minimum weighted vertex cover problem* (MWVC). Given a graph $G = (V, E)$, where the vertices $V = \{1, 2, \cdots, n\}$ are agents and each agent $i \in V$ has a weight c_i, we want to find a vertex set $V' \subseteq V$ such that for every edge $(u, v) \in E$ at least one of u and v is in V'. Here V' is called a vertex cover of G. The valuation of a vertex i is $-c_i$ if it is selected; otherwise its valuation is 0. We define $c(V') = \sum_{i \in V'} c_i$ for $V' \subseteq V$.

We want to find a vertex cover with the minimum weight, i.e., the objective function to be implemented is utilitarian. To use VCG mechanism, we need find the vertex cover with the minimum weight, which is NP-hard [14]. Since we are interested in the mechanisms that can be computed in polynomial time, we have to use some polynomial-time computable output functions. Many algorithms have been proposed in the literature to approximate the optimum solution. In this paper, we use a 2-approximation algorithm proposed in [14]. For the completeness of presentation, we briefly review their method here. The method is a round-based approach. Each round it selects some vertices and discards some vertices. For each vertex i, $w(i)$ is initialized to its weight c_i, and when $w(i)$ drops to 0, i is picked into the vertex cover. To make the presentation clear, we say an edge (i_1, j_1) is lexicographically smaller than edge (i_2, j_2) if (1) $\min(i_1, j_1) < \min(i_2, j_2)$, or (2) $\min(i_1, j_1) = \min(i_2, j_2)$ and $\max(i_1, j_1) < \max(i_2, j_2)$.

Algorithm 2 Approximate Minimum Weighted Vertex Cover

Input: A node weighted graph $G = (V, E, c)$.
Output: A vertex cover V'.

1: Set $V' = \emptyset$. For each $i \in V$, set $w(i) = c_i$.
2: **while** V' is not a vertex cover **do**
3: Pick an uncovered edge (i, j) with the least lexicographic order among all uncovered edges.
4: Let $m = \min(w(i), w(j))$.
5: Update $w(i)$ to $w(i) - m$ and $w(j)$ to $w(j) - m$.
6: If $w(i) = 0$, add i to V'. If $w(j) = 0$, add j to V'.
7: **end while**

Notice, selecting an edge using the lexicographic order does not necessarily achieve the 2-approximation ratio, but it guarantees the monotone property. The above algorithm produces a vertex cover V' whose weight is within 2 times of the optimum. For convenience, we use $\mathbf{VC}(c)$ to denote the vertex cover computed by Algorithm 2 when the cost vector of vertices is c.

We then generalize the above algorithm to a more general scenario. Consider a game $\mathcal{G}$ with setting $\mathcal{S}$, a set I of n agents and the cost vector c. Typically, a round-based output can be characterized as follows.

DEFINITION 2. *An updating rule $\mathcal{U}^r$ is said to be* crossing-independent *if, for any agent i not selected in round r, (1) $\mathcal{S}^{r+1}$ and c_{-i}^{r+1} do not depend on c_j^r (2) for fixed c_{-i}^r, $c_{i_1}^r \leq c_{i_2}^r$ implies that $c_{i_1}^{r+1} \leq c_{i_2}^{r+1}$.*

We have the following theorem about the existence of a truthful payment using $\mathcal{A}$.

THEOREM 8. *A round-based output $\mathcal{A}$, with the framework defined in Algorithm 3, satisfies MP if the output methods $\mathcal{O}^r$ satisfy MP and the updating function $\mathcal{U}^r$ is crossing-independent for every round r.*

If the round-based output satisfies monotone property, the cut-value always exists. We then show how to find the cut value for a selected agent k in Algorithm 4.

Algorithm 3 A General Round-Based Output $\mathcal{A}$

1: Set $r = 0$, $c^0 = c$, and $\mathcal{G}^0 = \mathcal{G}$ initially.
2: **repeat**
3: Compute an output o^r using a *deterministic* method

$$\mathcal{O}^r : \mathcal{S}^r \times c^r \to \{0,1\}^n.$$

Here $\mathcal{O}^r$, c^r and $\mathcal{S}^r$ are output function, cost vector and game setting in game $\mathcal{G}^r$ respectively. Notice $\mathcal{O}^r$ is often a simple greedy method such as selecting the agents that minimize some utilitarian function. For the example of vertex cover, $\mathcal{O}^r$ will always select the light-weighted node on the lexicographically least uncovered edge (i, j).
4: Let $r = r + 1$. Update the game, i.e., we get a new game $\mathcal{G}^r$ with setting $\mathcal{S}^r$ and cost vector c^r from $\mathcal{G}^{r-1}$ according to some rule

$$\mathcal{U}^r : o^{r-1} \times (\mathcal{S}^{r-1}, c^{r-1}) \to (\mathcal{S}^r, c^r).$$

Here we updates the cost and setting of the game. For the example of vertex cover, $\mathcal{S}^r$ is a new graph resulted from $\mathcal{S}^{r-1}$ by removing of all edges incident on the selected node; and the costs of nodes i, j are deduced by amount m. Remember here m is the minimum cost of i, and j in $\mathcal{G}^{r-1}$.
5: **until** a valid output is found
6: Return the union of selected players of all rounds as the final output. For the example of vertex cover, it is the union of nodes selected in all rounds.

Algorithm 4 Compute Cut Value for Round Based Method

Input: A round based output $\mathcal{A}$, the initial game $\mathcal{G}^1 = \mathcal{G}$ and updating function vector $\mathcal{U}$
Output: Cut value x for an agent k.

1: Set $r = 0$ and $c_k = \emptyset$. Here, $\emptyset$ is the value that can guarantee $\mathcal{A}_k = 0$.
2: **repeat**
3: Compute an output o^r using a deterministic method based on setting $\mathcal{S}^r$ using $\mathcal{O}^r : \mathcal{S}^r \times c^r \to \{0,1\}^n$.
4: Find the cut value for agent k based on output function $\mathcal{O}^r$ for costs c^r_{-k}. Let $\ell_r = \kappa_k(\mathcal{O}^r, c^r_{-k})$ be the cut value.
5: Set $r = r + 1$ and obtain the new game $\mathcal{G}_r$ from $\mathcal{G}^{r-1}$ and o^r according to the updating rule $\mathcal{U}^r$.
6: Let c^r be the new cost vector for game $\mathcal{G}^r$.
7: **until** a valid output is found.
8: Let $\overline{c}^1 = c|^k x$, and $(\mathcal{S}^r, \overline{c}^i) = \mathcal{U}^r(o^{r-1}, (\mathcal{S}^{r-1}, \overline{c}^{r-1}))$.
9: Find the minimum value x such that it satisfies the following inequations:

$$\begin{cases} x & \geq & \ell_1 \\ \overline{c}^1_i & \geq & \ell_2 \\ & \cdots & \\ \overline{c}^{r-1}_i & \geq & \ell_{r-1} \\ \overline{c}^r_i & \geq & \ell_r \end{cases}$$

10: Output the value x as the cut value.

The correctness of Algorithm 4 is straightforward. To compute the cut value, we assume that (1) we can solve the equation $\overline{c}^r_i = b$ to find x in polynomial time when the cost vector c_{-i} and b are given; (2) the cut value ℓ_i for each round i can be computed in polynomial time.

Now we consider the vertex cover problem. For each round r, we select a vertex with the least weight and that is incident on the lexicographically least uncovered edge. The output satisfies MP. For agent i, we update its cost to $c^r_i - c^r_j$ iff edge (i, j) is selected. It is easy to verify this updating rule is crossing-independent, thus we can apply Algorithm 4 to compute the cut value as shown in Algorithm 5.

Algorithm 5 Compute Cut Value for MVC.

Input: A node weighted graph $G = (V, E, c)$ and a node k selected by Algorithm 2.
Output: The cut value $\kappa_k(VC, c_{-k})$.

1: For each $i \in V$, set $w(i) = c_i$.
2: Set $w(k) = \infty$, $p_k = 0$ and $V' = \emptyset$.
3: **while** V' is not a vertex cover **do**
4: Pick an uncovered edge (i, j) with the least lexicographic order among all uncovered edges.
5: Set $m = \min(w(i), w(j))$.
6: Update $w(i) = w(i) - m$ and $w(j) = w(j) - m$.
7: If $w(i) = 0$, add i to V'; else add j to V'.
8: If $i == k$ or $j == k$ then set $p_k = p_k + m$.
9: **end while**
10: Output p_k as the cut value $\kappa_k(VC, c_{-k})$.

3.3.3 Complex Combinations

In section 3.3.1, we discussed how to find the cut value function when the output of the binary demand game is a simple combination of some outputs, whose cut values can be computed through other means (typically VCG). However, some algorithms cannot be decomposed in the way described in section 3.3.1.

Next we present a more complex way to combine functions, and as we may expected, the way to find the cut value is also more complicated. Assume that there are n agents $1 \leq i \leq n$ with cost vector c, and there are m binary demand games $\mathcal{G}_i$ with objective functions $f_i(o, c)$, setting $\mathcal{S}_i$ and output function ψ^i where $i = 1, 2, \cdots, m$. There is another binary demand game with setting $\mathcal{S}$ and output function $\mathcal{O}$, whose input is a cost vector $d = (d_1, d_2, \cdots, d_m)$. Let f be the function vector $(f_1, f_2, \cdots, f_m)$, ψ be the output function vector $(\psi^1, \psi^2, \cdots, \psi^m)$ and $\int$ be the setting vector $(\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_m)$. For notation simplicity, we define $\mathcal{F}_i(c) = f_i(\psi^i(c), c)$, for each $1 \leq i \leq m$; and define $\mathcal{F}(c) = (\mathcal{F}_1(c), \mathcal{F}_2(c), \cdots, \mathcal{F}_m(c))$.

Let us see a concrete example of these combinations. Consider a link weighted graph $G = (V, E, c)$, and a subset of q nodes $Q \subseteq$. The Steiner tree problem is to find a set of links with minimum total cost to connect Q. One way to find an approximation of the Steiner tree is as follows: (1) we build a virtual complete graph H using Q as its vertices, and the cost of each edge (i, j) is the cost of $\mathsf{LCP}(i, j, c)$ in graph G; (2) build the minimum spanning tree of H, denoted as $MST(H)$; (3) an edge of G is selected iff it is selected in some $\mathsf{LCP}(i, j, c)$ and edge (i, j) of H is selected to $MST(H)$.

In this game, we define $q(q-1)/2$ games $\mathcal{G}_{i,j}$, where $i, j \in Q$, with objective functions $f_{i,j}(o, c)$ being the minimum cost of connecting i and j in graph G, setting $\mathcal{S}_i$ being the original graph G and output function $\mathsf{LCP}(i, j, c)$. The game $\mathcal{G}$ corresponds to

the MST game on graph H. The cost of the pair-wise $q(q-1)/2$ shortest paths defines the input vector $d = (d_1, d_2, \cdots, d_m)$ for game MST. More details will be given in Section 4.2.

DEFINITION 3. *Given an output function $\mathcal{O}$ and setting $\mathcal{S}$, an objective function vector f, an output function vector ψ and setting vector $\int$, we define a compound binary demand game with setting $\mathcal{S}$ and output $\mathcal{O} \circ \mathcal{F}$ defined as $(\mathcal{O} \circ \mathcal{F})_i(c) = \bigvee_{j=1}^{m} (\mathcal{O}_j(\mathcal{F}(c)) \wedge \psi_i^j(c))$.*

The output method of the above definition can be interpreted as follows. An agent i is selected if and only if there is a j such that (1) i is selected in $\psi^j(c)$, and (2) the output method $\mathcal{O}$ will select index j under cost profile $\mathcal{F}(c)$. For simplicity, we will use $\mathcal{O} \circ \mathcal{F}$ to denote the output of this compound binary demand game.

Notice that a truthful payment scheme using $\mathcal{O} \circ \mathcal{F}$ as output exists if and only if it satisfies the monotone property. To study when $\mathcal{O} \circ \mathcal{F}$ satisfies MP, several necessary definitions are in order.

DEFINITION 4. **Function Monotone Property (FMP)** *Given an objective function g and an output method $\mathcal{O}$, a function $\mathcal{H}(c) = g(\mathcal{O}(c), c)$ is said to satisfy the function monotone property, if, given fixed c_{-i}, it satisfies:*

1. *When $\mathcal{O}_i(c) = 0$, $\mathcal{H}(c)$ does not increase over c_i.*
2. *When $\mathcal{O}_i(c) = 1$, $\mathcal{H}(c)$ does not decrease over c_i.*

DEFINITION 5. **Strong Monotone Property (SMP)** *An output method $\mathcal{O}$ is said to satisfy the strong monotone property if $\mathcal{O}$ satisfies MP, and for any agent i with $\mathcal{O}_i(c) = 1$ and agent $j \neq i$, $\mathcal{O}_i(c|^j c'_j) = 1$ if $c'_j \geq c_j$ or $\mathcal{O}_j(c|^j c'_j) = 0$.*

LEMMA 1. *For a given output function $\mathcal{O}$ satisfying SMP and cost vectors c, c' with $c_i = c'_i$, if $\mathcal{O}_i(c) = 1$ and $\mathcal{O}_i(c') = 0$, then there must exist $j \neq i$ such that $c'_j < c_j$ and $\mathcal{O}_j(c') = 1$.*

From the definition of the strong monotone property, we have Lemma 1 directly. We now can give a sufficient condition when $\mathcal{O} \circ \mathcal{F}$ satisfies the monotone property.

THEOREM 9. *If $\forall i \in [1, m]$, $\mathcal{F}_i$ satisfies FMP, ψ^i satisfies MP, and the output $\mathcal{O}$ satisfies SMP, then $\mathcal{O} \circ \mathcal{F}$ satisfies MP.*

See appendix for the proof. This theorem implies that there is a cut value for the compound output $\mathcal{O} \circ \mathcal{F}$. We then discuss how to find the cut value for this output. Below we will give an algorithm to calculate $\kappa_i(\mathcal{O} \circ \mathcal{F})$ when (1) $\mathcal{O}$ satisfies SMP, (2) ψ^j satisfies MP, and (3) for fixed c_{-i}, $\mathcal{F}_j(c)$ is a constant, say h_j, when $\psi_i^j(c) = 0$, and $\mathcal{F}_j(c)$ increases when $\psi_i^j(c) = 1$. Notice that here h_j can be easily computed by setting $c_i = \infty$ since ψ^j satisfies the monotone property. When given i and fixed c_{-i}, we define $(\mathcal{F}_j^i)^{-1}(y)$ as the smallest x such that $\mathcal{F}_j(c|^i x) = y$. For simplicity, we denote $(\mathcal{F}_j^i)^{-1}$ as $\mathcal{F}_j^{-1}$ if no confusion is caused when i is a fixed agent. In this paper, we assume that given any y, we can find such x in polynomial time.

THEOREM 10. *Algorithm 6 computes the correct cut value for agent i based on the output function $\mathcal{O} \circ \mathcal{F}$.*

See appendix for the proof. In most applications, the output function ψ^j implements the objective function f_j and f_j is utilitarian. Thus, we can compute the inverse of $\mathcal{F}_j^{-1}$ efficiently. Another issue is that it seems the conditions when we can apply Algorithm 6 are restrictive. However, lots of games in practice satisfies these properties and here we show how to deduct the MAX combination in [12]. For the completeness of our presentation, we first review the MAX combination first. Assume A_1 and A_2 are two allocation rules for single minded combinatorial auction, then the combination $MAX(A_1, A_2)$ returns the allocation with the larger welfare. If algorithm A_1 and A_2 satisfy MP and FMP, the operation $max(x, y)$ which returns the larger element of x and y satisfies SMP, from Theorem 9 we obtain that combination $MAX(A_1, A_2)$ also satisfies MP. Further, the cut value of the MAX combination can be found by Algorithm 6. As we will show in Section 4, the complex combination can apply to some more complicated problems.

Algorithm 6 Find Cut Value for Compound Method $\mathcal{O} \circ \mathcal{F}$

Input: Output function $\mathcal{O}$, objective function vector $\mathcal{F}$ and inverse function vector $\mathcal{F}^{-1} = \{\mathcal{F}_1^{-1}, \cdots, \mathcal{F}_m^{-1}\}$, output function vector ψ and fixed c_{-i}.
Output: Cut value for agent i based on $\mathcal{O} \circ \mathcal{F}$.

1: **for** $1 \leq j \leq m$ **do**
2: Compute the outputs $\psi^j(c_i)$.
3: Compute $h_j = \mathcal{F}_j(c|^i \infty)$.
4: **end for**
5: Use $h = (h_1, h_2, \cdots, h_m)$ as the input for the output function $\mathcal{O}$. Denote $\tau_j = \kappa_j(\mathcal{O}, h_{-j})$ as the cut value function of output $\mathcal{O}$ based on input h.
6: **for** $1 \leq j \leq m$ **do**
7: Set $\kappa_{i,j} = \mathcal{F}_j^{-1}(\min\{\tau_j, h_j\})$.
8: **end for**
9: The cut value for i is $\kappa_i(\mathcal{O} \circ \mathcal{F}, c_{-i}) = \max_{j=1}^{m} \kappa_{i,j}$.

4. CONCRETE EXAMPLES

4.1 Set Cover

In the set cover problem, there is a set U of m elements needed to be covered, and each agent $1 \leq i \leq n$ can cover a subset of elements S_i with a cost c_i. Let $S = \{S_1, S_2, \cdots, S_n\}$ and $c = (c_1, c_2, \cdots, c_n)$. We want to find a subset of agents D such that $U \subseteq \bigcup_{i \in D} S_i$. The selected subsets is called the set cover for U. The social efficiency of the output D is defined as $\sum_{i \in D} c_i$, which is the objective function to be minimized. Clearly, this is a utilitarian and thus VCG mechanism can be applied if we can find the subset of S that covers U with the minimum cost. It is well-known that finding the optimal solution is NP-hard. In [4], an algorithm of approximation ratio of H_m has been proposed and it has been proved that this is the best ratio possible for the set cover problem. For the completeness of presentation, we review their method here.

Algorithm 7 Greedy Set Cover (GSC)

Input: Agent i's subset S_i covered and cost c_i. $(1 \leq i \leq n)$.
Output: A set of agents that can cover all elements.

1: Initialize $r = 1$, $T_0 = \emptyset$, and $R = \emptyset$.
2: **while** $R \neq U$ **do**
3: Find the set S_j with the minimum density $\frac{c_j}{|S_j - T_r|}$.
4: Set $T_{r+1} = T_r \bigcup S_j$ and $R = R \bigcup j$.
5: $r = r + 1$
6: **end while**
7: Output R.

Let $GSC(S)$ be the sets selected by the Algorithm 7.Notice that

the output set is a function of S and c. Some works assume that the type of an agent could be c_i, i.e., S_i is assumed to be a public knowledge. Here, we consider a more general case in which the type of an agent is (S_i, c_i). In other words, we assume that every agent i can not only lie about its cost c_i but also can lie about the set S_i. This problem now looks similar to the combinatorial auction with single minded bidder studied in [11], but with the following differences: in the set cover problem we want to cover all the elements and the sets chosen can have some overlap while in combinatorial auction the chosen sets are disjoint.

We can show that the mechanism $M = (GSC, \mathcal{P}^{VCG})$, using Algorithm 7 to find a set cover and apply VCG mechanism to compute the payment to the selected agents, is not truthful. Obviously, the set cover problem is a binary demand game. For the moment, we assume that agent i won't be able to lie about S_i. We will drop this assumption later. We show how to design a truthful mechanism by applying our general framework.

1. **Check the monotone property**: The output of Algorithm 7 is a round-based output. Thus, for an agent i, we first focus on the output of one round r. In round r, if i is selected by Algorithm 7, then it has the minimum ratio $\frac{c_i}{|S_i - T_r|}$ among all remaining agents. Now consider the case when i lies its cost to $c_i' < c_i$, obviously $\frac{c_i'}{|S_i - T_r|}$ is still minimum among all remaining agents. Consequently, agent i is still selected in round r, which means the output of round r satisfies MP. Now we look into the updating rules. For every round, we only update the $T_{r+1} = T_r \bigcup S_j$ and $R = R \bigcup j$, which is obviously cross-independent. Thus, by applying Theorem 8, we know the output by Algorithm 7 satisfies MP.

2. **Find the cut value**: To calculate the cut value for agent i with fixed cost vector c_{-i}, we follow the steps in Algorithm 4. First, we set $c_i = \infty$ and apply Algorithm 7. Let i_r be the agent selected in round r and T_{r+1}^{-i} be the corresponding set. Then the cut value of round r is

$$\ell_r = \frac{c_{i_r}}{|S_{i_r} - T_r^{-i}|} \cdot |S_i - T_r^{-i}|.$$

Remember the updating rule only updates the game setting but not the cost of the agent, thus we have

$$\begin{cases} x \geq \ell_1 \\ \bar{c}_i^1 = x \geq \ell_2 \\ \cdots \\ \bar{c}_i^{t-1} = x \geq \ell_{t-1} \\ \bar{c}_i^t = x \geq \ell_t \end{cases}$$

Therefore, the final cut value for agent i is

$$\kappa_i(GSC, c_{-i}) = \max_r \{\frac{c_{i_r}}{|S_{i_r} - T_r^{-i}|} \cdot |S_i - T_r^{-i}|\}$$

The payment to an agent i is κ_i if i is selected; otherwise its payment is 0.

We now consider the scenario when agent i can lie about S_i. Assume that agent i cannot lie upward, i.e., it can only report a set $S_i' \subseteq S_i$. We argue that agent i will not lie about its elements S_i. Notice that the cut value computed for round r is $\ell^r = \frac{c_{i_r}}{|S_{i_r} - T_r^{-i}|} \cdot |S_i - T_r^{-i}|$. Obviously $|S_i' - T_r^{-i}| \leq |S_i - T_r^{-i}|$ for any $S_i' \subseteq S_i$. Thus, lying its set as S_i' will not increase the cut value for each round. Thus lying about S_i will not improve agent i's utility.

4.2 Link Weighted Steiner Trees

Consider any link weighted network $G = (V, E, c)$, where $E = \{e_1, e_2, \cdots, e_m\}$ are the set of links and c_i is the weight of the link e_i. The link weighted Steiner tree problem is to find a tree rooted at source node s spanning a given set of nodes $Q = \{q_1, q_2, \cdots, q_k\} \subset V$. For the simplicity of notations, we assume that $q_i = v_i$, for $1 \leq i \leq k$. Here the links are agents. The total cost of links in a graph $H \subseteq G$ is called the weight of H, denoted as $\omega(H)$. It is NP-hard to find the minimum cost multicast tree when given an arbitrary link weighted graph G [15, 16]. Takahashi and Matsuyama [16] first gave a polynomial-time 2-approximation algorithm for this problem. Then a series of results have been developed to improve the approximation ratio. The currently best polynomial time method has approximation ratio $1 + \frac{\ln 3}{2}$ [15]. Here, we review and discuss the method by Takahashi and Matsuyama [16].

Algorithm 8 Find LinkWeighted SteinerTree (LST)

Input: Network $G = (V, E, c)$ where c is the cost vector for link set E. Source node s and receiver set Q.
Output: A tree LST rooted at s and spanned all receivers.

1: Set $r = 1$, $G_1 = G$, $Q^1 = Q$ and $s^1 = s$.
2: **repeat**
3: In graph G_r, find the receiver, say q_i, that is closest to the source s, i.e., $\mathsf{LCP}(s, q_i, c)$ has the least cost among the shortest paths from s to all receivers in Q^r.
4: Select all links on $\mathsf{LCP}(s, q_i, c)$ as relay links and set their cost to 0. The new graph is denoted as G_{r+1}.
5: Set t_r as q_i and $\mathsf{P}_r = \mathsf{LCP}(s, q_i, c)$.
6: Set $Q^{r+1} = Q^r \backslash q_i$ and $r = r + 1$.
7: **until** all receivers are spanned.

Hereafter, let $LST(G)$ be the final tree constructed using the above method. It is shown in [20] that mechanism $M = (LST, p^{VCG})$ is not truthful, where p^{VCG} is the payment calculated based on VCG mechanism. We then show how to design a truthful payment scheme using our general framework. Observe that the output P_r, for any round r, satisfies MP, and the update rule for every round satisfies *crossing-independent*. Thus, from Theorem 8, the round-based output LST satisfies MP. In round r, the cut value for a link e_i can be obtained by using the VCG mechanism. Now we set $c_i = \infty$ and execute Algorithm 8. Let $w_r^{-i}(c_i)$ be the cost of the path $\mathsf{P}_r(c_i)$ selected in the rth round and $\Pi_r^i(c_i)$ be the shortest path selected in round r if the cost of c_i is temporarily set to $-\infty$. Then the cut value for round r is $\ell_r = w_r^i(c_{-i}) - |\Pi_r^i(c_{-i})|$ where $|\Pi_r^i(c_{-i})|$ is the cost of the path $\Pi_r^i(c_{-i})$ excluding node v_i. Using Algorithm 4, we obtain the final cut value for agent i: $\kappa_i(LST, c_{-i}) = \max_r\{\ell_r\}$. Thus, the payment to a link e_i is $\kappa_i(LST, c_{-i})$ if its reported cost is $d_i < \kappa_i(LST, d_{-i})$; otherwise, its payment is 0.

4.3 Virtual Minimal Spanning Trees

To connect the given set of receivers to the source node, besides the Steiner tree constructed by the algorithms described before, a virtual minimum spanning tree is also often used. Assume that Q is the set of receivers, including the sender. Assume that the nodes in a node-weighted graph are all agents. The virtual minimum spanning tree is constructed as follows.

The mechanism $M = (VMST, p^{VCG})$ is not truthful [20], where the payment p^{VCG} to a node is based on the VCG mechanism. We then show how to design a truthful mechanism based on the framework we described.

1. **Check the monotone property**: Remember that in the com-

Algorithm 9 Constructe VMST

1: **for** all pairs of receivers $q_i, q_j \in Q$ **do**
2: Calculate the least cost path $\mathsf{LCP}(q_i, q_j, d)$.
3: **end for**
4: Construct a virtual complete link weighted graph $K(d)$ using Q as its node set, where the link q_iq_j corresponds to the least cost path $\mathsf{LCP}(q_i, q_j, d)$, and its weight is $w(q_iq_j) = |\mathsf{LCP}(q_i, q_j, d)|$.
5: Build the minimum spanning tree on $K(d)$, denoted as $VMST(d)$.
6: **for** every virtual link q_iq_j in $VMST(d)$ **do**
7: Find the corresponding least cost path $\mathsf{LCP}(q_i, q_j, d)$ in the original network.
8: Mark the agents on $\mathsf{LCP}(q_i, q_j, d)$ selected.
9: **end for**

plete graph $K(d)$, the weight of a link q_iq_j is $|\mathsf{LCP}(q_i, q_j, d)|$. In other words, we implicitly defined $|Q|(|Q|-1)/2$ functions $f_{i,j}$, for all $i < j$ and $q_i \in Q$ and $q_j \in Q$, with $f_{i,j}(d) = |\mathsf{LCP}(q_i, q_j, d)|$. We can show that the function $f_{i,j}(d) = |\mathsf{LCP}(q_i, q_j, d)|$ satisfies FMP, LCP satisfies MP, and the output MST satisfies SMP. From Theorem 9, the output method VMST satisfies the monotone property.

2. **Find the cut value**: Notice VMST is the combination of MST and function $f_{i,j}$, so cut value for VMST can be computed based on Algorithm 6 as follows.

   (a) Given a link weighted complete graph $K(d)$ on Q, we should find the cut value function for edge $e_k = (q_i, q_j)$ based on MST. Given a spanning tree T and a pair of terminals p and q, clearly there is a unique path connecting them on T. We denote this path as $\Pi_T(p, q)$, and the edge with the maximum length on this path as $LE(p, q, T)$. Thus, the cut value can be represented as $\kappa_k(MST, d) = LE(q_i, q_j, MST(d|^k\infty))$

   (b) We find the value-cost function for LCP. Assume $v_k \in \mathsf{LCP}(q_i, q_j, d)$, then the value-cost function is $x_k = y_k - |\mathsf{LCP}_{v_k}(q_i, q_j, d|^k 0)|$. Here, $\mathsf{LCP}_{v_k}(q_i, q_j, d)$ is the least cost path between q_i and q_j with node v_k on this path.

   (c) Remove v_k and calculate the value $K(d|^k\infty)$. Set $h_{(i,j)} = |\mathsf{LCP}(q_i, q_j, d|^\infty))|$ for every pair of node $i \neq j$ and let $h = \{h_{(i,j)}\}$ be the vector. Then it is easy to show that $\tau_{(i,j)} = |LE(q_i, q_j, MST(h|^{(i,j)}\infty))|$ is the cut value for output VMST. It easy to verify that $\min\{h_{(i,j)}, \tau_{(i,j)}\} = |LE(q_i, q_j, MST(h)|$. Thus, we know $\kappa_k^{(i,j)}(VMST, d)$ is $|LE(q_i, q_j, MST(h)| - |\mathsf{LCP}_{v_k}(q_i, q_j, d|^k 0)|$. The cut value for agent k is $\kappa_k(VMST, d_{-k}) = \max_{0\le i,j\le r} \kappa_k^{ij}(VMST, d_{-k})$.

3. We pay agent k $\kappa_k(VMST, d_{-k})$ if and only if k is selected in $VMST(d)$; else we pay it 0.

4.4 Combinatorial Auctions

Lehmann *et al.* [11] studied how to design an efficient truthful mechanism for single-minded combinatorial auction. In a single-minded combinatorial auction, there is a set of items S to be sold and there is a set of agents $1 \le i \le n$ who wants to buy some of the items: agent i wants to buy a subset $S_i \subseteq S$ with maximum price m_i. A single-minded bidder i declares a bid $b_i = \langle S_i', a_i\rangle$ with $S_i' \subseteq S$ and $a_i \in R^+$. Two bids $\langle S_i', a_i\rangle$ and $\langle S_j', a_j\rangle$ conflict if $S_i' \cap S_j' \neq \emptyset$. Given the bids $b_1, b_2, \cdots, b_n$, they gave a greedy round-based algorithm as follows. First the bids are sorted by some criterion ($\frac{a_i}{|S_i'|^{1/2}}$ is used in[11]) in an increasing order and let L be the list of sorted bids. The first bid is granted. Then the algorithm exams each bid of L in order and grants the bid if it does not conflict with any of the bids previously granted. If it does, it is denied. They proved that this greedy allocation scheme using criterion $\frac{a_i}{|S_i'|^{1/2}}$ approximates the optimal allocation within a factor of $\sqrt{m}$, where m is the number of goods in S.

In the auction settings, we have $c_i = -a_i$. It is easy to verify the output of the greedy algorithm is a round-based output. Remember after bidder j is selected for round r, every bidder has conflict with j will not be selected in the rounds after. This equals to update the cost of every bidder having conflict with j to 0, which satisfies *crossing-independent*. In addition, in any round, if bidder i is selected with a_i then it will still be selected when it declares $a_i' > a_i$. Thus, for every round, it satisfies MP and the cut value is $|S_i'|^{1/2} \cdot \frac{a_{j_r}}{|S_{j_r}{}'|^{1/2}}$ where j_r is the bidder selected in round r if we did not consider the agent i at all. Notice $\frac{a_{jr}}{|S_{jr}{}'|^{1/2}}$ does not increase when round r increases, so the final cut value is $|S_i'|^{1/2} \cdot \frac{a_j}{|S_j{}'|^{1/2}}$ where b_j is the *first* bid that has been denied but would have been selected were it not only for the presence of bidder i. Thus, the payment by agent i is $|S_i'|^{1/2} \cdot \frac{a_j}{|S_j{}'|^{1/2}}$ if $a_i \ge |S_i'|^{1/2} \cdot \frac{a_j}{|S_j{}'|^{1/2}}$, and 0 otherwise. This payment scheme is exactly the same as the payment scheme in [11].

5. CONCLUSIONS

In this paper, we studied how to design a truthful mechanism $M = (\mathcal{O}, \mathcal{P})$ using a given output method $\mathcal{O}$. We showed that the output method $\mathcal{O}$ satisfying the monotone property is the necessary and sufficient condition such that a truthful mechanism M exists. By considering a polynomial-time computable output method $\mathcal{O}$, we then studied in detail how to compute the payment $\mathcal{P}$ efficiently such that the mechanism $M = (\mathcal{O}, \mathcal{P})$ is computable in polynomial time. Our main contribution here is that we presented several general methods to compute the payment efficiently for various monotone output methods $\mathcal{O}$, such as or/and combination, the round-based algorithms, and the composition-based algorithms. Several concrete examples are studied to demonstrate the effectiveness of our methods of computing the payments in polynomial time.

Computational Complexity and Approximation Ratios: In this paper, we mainly concentrated on how to compute the payments in polynomial time. We have to point it out that our method is not necessarily the time-optimal for computing the payments for mechanism $M = (\mathcal{O}, \mathcal{P})$ when a given monotone output $\mathcal{O}$ is used. It seems impossible to design a general framework for computing the payment with optimal time complexity without studying the specifics of the original binary selection problem and the output method $\mathcal{O}$. We have made some progress in this latter direction. We showed in [18] a method to compute the payment for unicast in a node weighted graph in optimal time complexity $O(n \log n + m)$.

Another interesting research direction is, given a optimization binary demand game, how to design an approximation method $\mathcal{O}$ satisfying the monotone property with good approximation ratio. Many works [12, 11] in the mechanism design literature are in this direction. We pointed out here that the attempt of this paper is not to design a better output method for a problem, but to design a method to compute the payment efficiently when $\mathcal{O}$ is given. Thus,

it would be interesting to find the output algorithm with the good approximation ratio such that it has a computable payment scheme.

Beyond Binary Demand Game: In this paper, we studied mechanism design for binary demand games. However, some problems cannot be directly formulated as binary demand game. The job-scheduling problem $(Q||C_{\max})$ is such an example. This has been studied extensively in [2]. It is known that a truthful payment scheme $\mathcal{P}$ exists for an output method $\mathcal{O}$ if and only if the workload w_i assigned by $\mathcal{O}$ is non-decreasing in s_i when fixing s_{-i}.

Theorem 4 can be extended to a general output method $\mathcal{O}$, whose range is R^+. The remaining difficulty is then how to compute the payment $\mathcal{P}$ under mild assumptions about the valuations if a truthful mechanism $M = (\mathcal{O}, \mathcal{P})$ does exist.

6. REFERENCES

[1] A. ARCHER, C. PAPADIMITRIOU, K. T., AND TARDOS, E. An approximate truthful mechanism for combinatorial auctions with single parameter agents. In *Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete algorithms* (2003), Society for Industrial and Applied Mathematics, pp. 205–214.

[2] ARCHER, A., AND TARDOS, E. Truthful mechanisms for one-parameter agents. In *Proceedings of the 42nd IEEE symposium on Foundations of Computer Science* (2001), IEEE Computer Society, p. 482.

[3] AULETTA, V., PRISCO, R. D., PENNA, P., AND PERSIANO, P. Deterministic truthful approximation schemes for scheduling related machines.

[4] CHVATAL, V. A greedy heuristic for the set covering problem. *Mathematics of Operations Research 4*, 3 (1979), 233–235.

[5] CLARKE, E. H. Multipart pricing of public goods. *Public Choice* (1971), 17–33.

[6] DEVANUR, N. R., MIHAIL, M., AND VAZIRANI, V. V. Strategyproof cost-sharing mechanisms for set cover and facility location games. In *ACM Electronic Commerce (EC03)* (2003).

[7] FEIGENBAUM, J., KRISHNAMURTHY, A., SAMI, R., AND SHENKER, S. Approximation and collusion in multicast cost sharing (abstract). In *ACM Economic Conference* (2001).

[8] FEIGENBAUM, J., PAPADIMITRIOU, C., SAMI, R., AND SHENKER, S. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing.* (2002), pp. 173–182.

[9] GREEN, J., AND LAFFONT, J. J. Characterization of satisfactory mechanisms for the revelation of preferences for public goods. *Econometrica* (1977), 427–438.

[10] GROVES, T. Incentives in teams. *Econometrica* (1973), 617–631.

[11] LEHMANN, D., OCALLAGHAN, L. I., AND SHOHAM, Y. Truth revelation in approximately efficient combinatorial auctions. *Journal of ACM 49*, 5 (2002), 577–602.

[12] MU'ALEM, A., AND NISAN, N. Truthful approximation mechanisms for restricted combinatorial auctions: extended abstract. In *18th National Conference on Artificial intelligence* (2002), American Association for Artificial Intelligence, pp. 379–384.

[13] NISAN, N., AND RONEN, A. Algorithmic mechanism design. In *Proc. 31st Annual Symposium on Theory of Computing (STOC99)* (1999), pp. 129–140.

[14] R. BAR-YEHUDA, R, S. E. *A local-ratio theorem for approximating the weighted vertex cover problem*, vol. 25. Elsevier science publishing company, Amsterdam, 1985.

[15] ROBINS, G., AND ZELIKOVSKY, A. Improved steiner tree approximation in graphs. In *Proceedings of the 11th annual ACM-SIAM symposium on Discrete algorithms* (2000), Society for Industrial and Applied Mathematics, pp. 770–779.

[16] TAKAHASHI, H., AND MATSUYAMA, A. An approximate solution for the steiner problem in graphs. *Math. Japonica 24* (1980), 573–577.

[17] VICKREY, W. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance* (1961), 8–37.

[18] WANG, W., AND LI, X.-Y. Truthful low-cost unicast in selfish wireless networks. In *4th International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks (WMAN) of IPDPS* (2004).

[19] WANG, W., LI, X.-Y., AND SUN, Z. Design multicast protocols for non-cooperative networks, 2004. Submitted for publication.

[20] WANG, W., LI, X.-Y., AND WANG, Y. Truthful multicast in selfish wireless networks. Accepted for publication.

7. APPENDIX

The output o^{min} that minimizes the objective function $g(o, c)$ depends on the valuation vector v and the objective function g itself, so we will use $o^{\min}(g, c)$ to denote o^{min}. In a binary demand game with an objective function $g(o, c)$, if we fix the output $o_i = 1$, then the objective function is denoted as $g((o_{-i}, 1), c)$; if we fix the output $o_i = 0$, the objective function is denoted as $g((o_{-i}, 0), c)$.

Theorem 1: If a mechanism $M = (\mathcal{O}, \mathcal{P})$ satisfies IC, then $\forall i$, if $\mathcal{O}_i(t|^i t_{i_1}) = \mathcal{O}_i(t|^i t_{i_2})$, then $p_i(t|^i t_{i_1}) = p_i(t|^i t_{i_2})$.

PROOF. We prove it by contradiction. Without loss of generality, assume $p_i(t|^i t_{i_1}) > p_i(t|^i t_{i_2})$. Now consider a profile $t' = t|^i t_{i_2}$. When agent i declares v_{i_1}, its utility is

$$\begin{aligned} u_i(t|^i t_{i_1}) &= p_i(t|^i t_{i_1}) + v(\mathcal{O}_i(t|^i t_{i_1}), t_{i_2}) \\ &= p_i(t|^i t_{i_1}) + v(\mathcal{O}_i(t|^i t_{i_2}), t_{i_2}) \\ &> p_i(t|^i t_{i_2}) + v(\mathcal{O}_i(t|^i t_{i_2}), t_{i_2}) \\ &= u_i(t|^i t_{i_2}) \end{aligned}$$

The above inequality implies that by lying its valuation to t_{i_1}, agent i could benefit, which is a contradiction. □

Theorem 4 For any agent i and any fixed c_{-i} in a binary demand game with output method $\mathcal{O}$, the following three conditions are equivalent:

1. There exists a value $\kappa_i(\mathcal{O}, c_{-i})$ (called a *cut value*) such that $\mathcal{O}_i(c) = 1$ if $c_i < \kappa_i(\mathcal{O}, c_{-i})$ and $\mathcal{O}_i(c) = 0$ if $c_i > \kappa_i(\mathcal{O}, c_{-i})$. When $c_i = \kappa_i(\mathcal{O}, c_{-i})$, $\mathcal{O}_i(c)$ could be either 0 or 1, depending on the tie-breaker of the output method $\mathcal{O}$. Hereafter, we will not consider the tie-breaker scenario in our proofs.
2. The output method $\mathcal{O}$ satisfies MP.
3. There exists a truthful payment schemes $\mathcal{P}$ for this binary demand game.

PROOF. We break the proof into three directions as follow.

Direction 1:CONDITION 2 IMPLIES CONDITION 1.

The proof of this direction is straightforward and is omitted here.

Direction 2:CONDITION 3 IMPLIES CONDITION 2.

The proof of this direction is similar to a proof in [12]. To prove this direction, we assume there exists an agent i and two valuation vectors $c|^i c_{i_1}$ and $c|^i c_{i_2}$, where $c_{i_1} < c_{i_2}$, $\mathcal{O}_i(c|^i c_{i_2}) = 1$ and $\mathcal{O}_i(c|^i c_{i_1}) = 0$. From corollary 2, we know that $p_i(c|^i c_{i_1}) = p_i^0$ and $p_i(c|^i c_{i_2}) = p_i^1$.

Now fix c_{-i}, the utility for i when $c_i = c_{i_1}$ is $u_i(c_{i_1}) = p_i^0$. When agent i lies its valuation to c_{i_2}, its utility is $p_i^1 - c_{i_1}$. Since $M = (\mathcal{O}, \mathcal{P})$ is truthful, we have

$$p_i^0 > p_i^1 - c_{i_1} \tag{1}$$

Now consider the scenario when the actual valuation of agent i is $c_i = c_{i_2}$. Its utility is $p_i^1 - c_{i_2}$ when it reports its true valuation. Similarly, if it lies its valuation to c_{i_1}, its utility is p_i^0. Since $M = (\mathcal{O}, \mathcal{P})$ is truthful, we have

$$p_i^0 < p_i^1 - c_{i_2} \tag{2}$$

Combining inequalities (1) and (2), we have $p_i^1 - c_{i_2} > p_i^0 > p_i^1 - c_{i_1}$. This inequality implies that $c_{i_1} > c_{i_2}$, which is a contradiction.

Direction 2:CONDITION 1 IMPLIES CONDITION 3.

We prove this direction by constructing a payment scheme and proving that this payment scheme is truthful. The payment scheme is: If $\mathcal{O}_i(c) = 1$, then agent i gets payment $p_i(c) = \kappa_i(\mathcal{O}, c_{-i})$; else it gets payment $p_i(c) = 0$.

From condition 1, if $\mathcal{O}_i(c) = 1$ then $c_i > \kappa_i(\mathcal{O}, c_{-i})$. Thus, its utility is $\kappa_i(\mathcal{O}, c_{-i}) - c_i > 0$, which implies that the payment scheme satisfies the IR. In the following we prove that this payment scheme also satisfies IC property. There are two cases here.

Case 1: $c_i < \kappa(\mathcal{O}, c_{-i})$. In this case, when i declares its true cost c_i, its utility is $\kappa_i(\mathcal{O}, c_{-i}) - c_i > 0$. Now consider the situation when i declares a cost $d_i \neq c_i$. If $d_i < \kappa_i(\mathcal{O}, c_{-i})$, then i gets the same payment and utility since it is still selected. If $d_i > \kappa_i(\mathcal{O}, c_{-i})$, then its utility becomes 0 since it is not selected anymore. Thus, it has no incentive to lie in this case.

Case 2: $c_i \geq \kappa(\mathcal{O}, c_{-i})$. In this case, when i reveals its true valuation, its payment is 0 and the utility is 0. Now consider the situation when i declares a valuation $d_i \neq c_i$. If $d_i > \kappa_i(\mathcal{O}, c_{-i})$, then i gets the same payment and utility since it is still not selected. If $d_i \leq \kappa_i(\mathcal{O}, c_{-i})$, then its utility becomes $\kappa_i(\mathcal{O}, c_{-i}) - c_i \leq 0$ since it is selected now. Thus, it has no incentive to lie in this case. □

Theorem 5: The payment defined Alforithm 1 is minimum among all truthful payment schemes using $\mathcal{O}$ as output.

PROOF. Assume that there is a truthful payment scheme p' pays agent i a payment $\kappa(\mathcal{O}, c_{-i}) - \epsilon$, for some $\epsilon > 0$. Notice that the payment to an agent i does not depend on its actual cost as long as $\mathcal{O}_i$ does not change. When agent i has cost $\kappa(\mathcal{O}, c_{-i}) - \epsilon/2$, it is still selected. Thus, agent i still gets payment $\kappa(\mathcal{O}, c_{-i}) - \epsilon$ under scheme p'. Consequently, the utility of agent i becomes $-\epsilon/2$, which violates the individual rationality property. This finishes the proof. □

Theorem 8: A round-based output $\mathcal{A}$, with the framework defined in Algorithm 3, satisfies MP if the output methods $\mathcal{O}^r$ satisfy MP and the updating function $\mathcal{U}^r$ is crossing-independent for every round r.

PROOF. Consider an agent i and fixed c_{-i}. We prove that when an agent i is selected with cost c_i, then it is also selected with cost $d_i < c_i$. Assume that i is selected in round r with cost c_i. Then under cost d_i, if agent i is selected in a round before r, our claim holds. Otherwise, consider in round r. Clearly, the setting $\mathcal{S}^r$ and the costs of all other agents are the same as what if agent i had cost c_i since i is not selected in the previous rounds due to the cross-independent property. Since i is selected in round r with cost c_i, i is also selected in round r with $d_i < c_i$ due to the reason that $\mathcal{O}^r$ satisfies MP. This finishes the proof. □

Theorem 9 Assume that for every $1 \leq i \leq m$, $\mathcal{F}_i$ satisfies FMP, ψ^i satisfies MP, and the output method $\mathcal{O}$ satisfies SMP. Then $\mathcal{O} \circ \mathcal{F}$ satisfies MP.

PROOF. Assuming for cost vector c we have $(\mathcal{O} \circ \mathcal{F})_i(c) = 1$, we should prove for any cost vector $c' = c|^i c_i'$ with $c_i' < c_i$, $(\mathcal{O} \circ \mathcal{F})_i(c') = 1$. Noticing that $(\mathcal{O} \circ \mathcal{F})_i(c) = 1$, without loss of generality, we assume that $\mathcal{O}_k(\mathcal{F}(c)) = 1$ and $\psi_i^k(c) = 1$ for some index $1 \leq k \leq m$.

Now consider the output $\mathcal{O}$ with the cost vector $\mathcal{F}(c')|^k \mathcal{F}_k(c)$. There are two scenarios, which will be studied one by one as follows.

One scenario is that index k is not chosen by the output function $\mathcal{O}$. From Lemma 1, there must exist $j \neq k$ such that

$$\mathcal{F}_j(c') < \mathcal{F}_j(c) \tag{3}$$

and

$$\mathcal{O}_j(\mathcal{F}(c')|^k \mathcal{F}_k(c)) = 1 \tag{4}$$

We then prove that agent i will be selected in the output $\psi^j(c')$, i.e., $\psi_i^j(c') = 1$. If it is not, since $\psi^j(c)$ satisfies MP, we have $\psi_i^j(c) = \psi_i^j(c') = 0$ from $c_i' < c_i$. Since $\mathcal{F}_j$ satisfies FMP, we know $\mathcal{F}_j(c') \geq \mathcal{F}_j(c)$, which is a contradiction to the inequality (3). Consequently, we have $\psi_i^j(c') = 1$. From Equation (4), the fact that index k is not selected by output function $\mathcal{O}$ and the definition of SMP, we have

$$\mathcal{O}_j(\mathcal{F}(c')) = 1,$$

Thus, agent i is selected by $\mathcal{O} \circ \mathcal{F}$ because of $\mathcal{O}_j(\mathcal{F}(c')) = 1$ and $\psi_i^j(c') = 1$.

The other scenario is that index k is chosen by the output function $\mathcal{O}$. First, agent i is chosen in $\psi^k(c')$ since the output $\psi^k(c)$ satisfies the monotone property and $c_i' < c_i$ and $\psi_i^k(c) = 1$. Secondly, since the function $\mathcal{F}_k$ satisfies FMP, we know that $\mathcal{F}_k(c') \leq \mathcal{F}_k(c)$. Remember that output $\mathcal{O}$ satisfies the SMP, thus we can obtain $\mathcal{O}_k(\mathcal{F}(c')) = 1$ from the fact that $\mathcal{O}_k(\mathcal{F}(c')|^k\mathcal{F}_k(c)) = 1$ and $\mathcal{F}_k(c') \leq \mathcal{F}_k(c)$. Consequently, agent i will also be selected in the final output $\mathcal{O} \circ \mathcal{F}$. This finishes our proof. □

Theorem 10 Algorithm 6 computes the correct cut value for agent i based on the output function $\mathcal{O} \circ \mathcal{F}$.

PROOF. In order to prove the correctness of the cut value function calculated by Algorithm 6, we prove the following two cases. For our convenience, we will use κ_i to represent $\kappa_i(\mathcal{O} \circ \mathcal{F}, c_{-i})$ if no confusion caused.

First, if $d_i < \kappa_i$ then $(\mathcal{O} \circ \mathcal{F})_i(c|^i d_i) = 1$. Without loss of generality, we assume that $\kappa_i = \kappa_{i,j}$ for some j. Since function $\mathcal{F}_j$ satisfies FMP and $\psi_i^j(c|^i d_i) = 1$, we have $\mathcal{F}_j(c|^i d_i) < \mathcal{F}_j(\kappa_i)$. Notice $d_i < \kappa_{i,j}$, from the definition of $\kappa_{i,j} = \mathcal{F}_j^{-1}(\min\{\tau_j, h_j\})$ we have (1) $\psi_i^j(c|^i d_i) = 1$, (2) $\mathcal{F}_j(c|^i d_i) < \tau_j$ due to the fact that $\mathcal{F}_j(x)$ is a non-decreasing function when j is selected. Thus, from the monotone property of $\mathcal{O}$ and τ_j is the cut value for output $\mathcal{O}$, we have

$$\mathcal{O}_j(h|^j\mathcal{F}_j(c|^i d_i)) = 1. \tag{5}$$

If $\mathcal{O}_j(\mathcal{F}(c|^i d_i)) = 1$ then $(\mathcal{O}\circ\mathcal{F})_i(c|^i d_i) = 1$. Otherwise, since $\mathcal{O}$ satisfies SMP, Lemma 1 and equation 5 imply that there exists at least one index k such that $\mathcal{O}_k(\mathcal{F}(c|^i d_i)) = 1$ and $\mathcal{F}_k(c|^i d_i) < h_k$. Note $\mathcal{F}_k(c|^i d_i) < h_k$ implies that i is selected in $\psi^k(c|^i d_i)$ since $h_k = \mathcal{F}_k(c_i|^i\infty)$. In other words, agent i is selected in $\mathcal{O}\circ\mathcal{F}$.

Second, if $d_i \geq \kappa_i(\mathcal{O} \circ \mathcal{F}, c_{-i})$ then $(\mathcal{O} \circ \mathcal{F})_i(c|^i d_i) = 0$. Assume for the sake of contradiction that $(\mathcal{O} \circ \mathcal{F})_i(c|^i d_i) = 1$. Then there exists an index $1 \leq j \leq m$ such that $\mathcal{O}_j(\mathcal{F}(c|^i d_i)) = 1$ and $\psi_i^j(c|^i d_i) = 1$. Remember that $h_k \geq \mathcal{F}_k(c|^i d_i)$ for any k. Thus, from the fact that $\mathcal{O}$ satisfies SMP, when changing the cost vector from $\mathcal{F}(c|^i d_i)$ to $h|^j\mathcal{F}_j(c|^i d_i)$, we still have $\mathcal{O}_j(h|^j\mathcal{F}_j(c|^i d_i)) = 1$. This implies that

$$\mathcal{F}_j(c|^i d_i) < \tau_j.$$

Combining the above inequality and the fact that $\mathcal{F}_j(c|^i c|^i d_i) < h_j$, we have $\mathcal{F}_j(c|^i d_i) < \min\{h_j, \tau_j\}$. This implies

$$d_i < \mathcal{F}_j^{-1}(\min\{h_j, \tau_j\}) = \kappa_{i,j} < \kappa_i(\mathcal{O} \circ \mathcal{F}, c_{-i}).$$

which is a contradiction. This finishes our proof. □