The Role of LP Duality in Computational Mechanism Design

Dinesh Garg
Computer Science and Automation
Indian Institute of Science
Bangalore - 560 012

Abstract— **An important fundamental problem in e-commerce is to design a mechanism that computes optimal system-wide solutions despite the self-interest of individual users (buyers, sellers, brokers, etc.) and computational agents (buying agents, selling agents, etc.). This class of problems, formally called computational mechanism design problems, are similar in nature to the economic mechanism design problems arising in economics and game theory. Classic game-theoretic solutions for these problems are often prohibitively expensive computationally. For example, the Generalized Vickrey Auction (GVA) is a classical mechanism to solve the Combinatorial Allocation Problem (CAP). Despite its rich game theoretic properties, the GVA mechanism is impractical from a computational point of view as it presents a number of computational challenges starting from the NP-hard winner determination problem through hard valuation problem of the agents, to prohibitively high communication costs. A plethora of proposals has emerged in recent times to address these problems. One of the popular approaches proposed is based on using the primal-dual algorithm for solving the mechanism design problem.**

The objective of this paper is two fold. First, we bring out the challenges and difficulties in adopting the classical game-theoretic solutions to computational mechanism design problems. Next we show that LP duality can play a useful role in computational mechanism design. We discuss a representative algorithm based on duality, ”COMBAUCTION” for solving the combinatorial allocation problem.

Keywords

Economic Mechanism Design (EMD), Computational Mechanism Design (CMD), Game Theory, Combinatorial Allocation Problem (CAP), VCG (Vickrey-Clarkes-Groves) Mechanisms, Generalized Vickrey Auction (GVA), Primal Dual Algorithm.

I. INTRODUCTION

In the realm of computational and information sciences, the topic of *Computational Mechanism Design (CMD)* has received considerable attention in ithe past few years. The Internet has given rise to a new paradigm namely distributed open computer networks, in which agents- users and computational devices- are not cooperative, but self-interested, with private information and goal. The examples include: (a) Network routing problem with self-interest packets and routers; (b) procurement problems in e-commerce between selfish buyer and sellers; (c) logistics problem, with task allocation across multiple self-interested shipping companies. A fundamental problem in building these systems is to design incentive-compatible protocols, which compute optimal system-wide solution despite the self-interest of individual agents. This emerging area of study is known as computational mechanism design [1].

Microeconomics and game theory are two mature disciplines for dealing the similar kind of situations in human society. These science have proved able to coordinate the activities of many autonomous individuals in human societies. Therefore, it is natural to look if a parallel can be drawn between classical Economic Mechanism Design (EMD)[2] [3], also called Game-Theoretic Mechanism Design, and CMD. In otherwords, it is obvious to seek for a design methodology for CMD which can reinforce the design by adding desirable game-theoretic properties in it.

Exploration of the interface between EMD and CMD has established the fact that one can always achieve the desirable game theoretic properties in the design of computational mechanisms by importing the design principles from EMD; but at the cost of deep computational requirement [1]. Therefore, the current research lies at resolving the tension between what one might choose to do game-theoretically and what is desirable computationally. This stringent requirement of deep computation can be stated formally as below:

”In most of the cases it turns out that the agents and mechanism needs to solve a number of NP-Hard problems if the overall design has to be reinforced by achieving desirable game-theoretic properties.”

Such problems have been classified systematically into two broad categories and then further into four sub-categories.

(1) At Agent Level:

(A) Valuation Complexity: How much computation is required for an agent to provide information about its preference to the mechanism ? [1][4]
(B) Strategic Complexity: How much computation an agent needs to model the strategy of other agents and then solve a game-theoretic problem to compute its own strategy.

(2) At the Infrastructure Level:

(A) Winner-determination Complexity: How much computation is expected for the mechanism to compute an outcome given information provided by agents?
(B) Communication Complexity: How much communication is required, between agents and mechanism, to compute an outcome?

Each one of these problems has given rise to a separate direction for the current research and a number of proposal exist to address the problem. Resolving the problem of valuation complexity is a current popular research topic and also the focus of this paper. In this paper we consider Combinatorial Allocation Problem (CAP) which is a proved NP-Hard problem. The Generalized Vickrey Auction (GVA) Mechanism [5] is a classical mechanism which solves the CAP by computing vickrey payments for agents. GVA also has two useful game-theoretic properties namely Allocative Efficiency and Strategy Proofness. It turns out that $(I+1)$ NP-Hard problems need to be solved at the infrastructure level and an exponential number of hard valuation problems at each agents level, when the CAP is tried to solved by straightforward implementation of the GVA. Here I represents the number of agents in the system. The hard valuation for agents in solving the CAP is the prime motive for the Ph.D. work of David C. Parkes and also of this paper [1].

It was the paper by Bertsekas [6] which first suggested a primal-dual algorithm for a combinatorial optimization problem, in particular, the assignment problem - a special case of the CAP- but without having worried about strategy proofness property. Inspiring from this Parkes has come up with a primal dual algorithm, called **COMBAUCTION**, for solving the CAP and also its extension, called **VICKAUCTION**, to compute vickrey prices. The algorithm essentially computes the solution to linear program formulation of CAP, which is integer linear program otherwise, proposed by Bickchandani & Ostroy [7]. The algorithm can be given a natural interpretation as an iterative combinatorial auction. These algorithms provide an interactive solution, requiring *just enough* information from agents to compute the efficient allocation and vickrey prices.

A. Outline

The paper is organized in the following way. In Section 2, we delineate the Computational Mechanism Design problem. We also provide a real world example to motivate the problem. Section 3 is devoted to introduction to *Classical Theory of Mechanism Design*. First, we provide game theoretic and economic perspective of Mechanism Design. Next, we discuss a few game-theoretic properties which an ideal mechanism should have. We also describe briefly the revelation principle, which has proved a powerful theoretical tool in mechanism design theory, and introduce incentive-compatible and direct revelation mechanisms. We conclude this section by providing a brief description about Vickrey-Clarke-Groves mechanisms, including GVA mechanism for CAP problem. Section 4 considers the different computational concerns in an implemented mechanism, looking at computation both at the agent and at the mechanism infrastructure level. We consider the consequences of the revelation principle for CMD. This section provides the motivation for the actual problem which we are going to address in this paper i.e. role of duality in Computational Mechanism Design. Section 5 describes the Combinatorial Allocation Problem and LP formulation for the same as proposed by Bickchandani and Ostroy [7]. Section 6 provides background on linear programming theory and primal dual algorithm. Section 7 describes the COMBAUCTION - a primal dual algorithm for the CAP discussed by David Parkes [1]. Section 8 constitutes conclusions and outlines some directions for future work.

II. Computational Mechanism Design

As mentioned earlier that the Internet embodies a new paradigm of distributed open computer networks, in which agents-users and computational devices-are (1)Non Cooperative, (2) Self Interested, (3) Rational, (4)Have their private information about rest of the world (other agent's goal and type), (5) Have their own set of goals. Collectively these facts can be called as agent's type, θ. A fundamental problem in building these systems is to design *incentive compatible* protocols, which compute optimal system wide solution despite the self interest of individual agents.

We can view these problems as distributed optimization problems, with an objective function that depends on the *private information* of the agents in the system. A rational self-interested agents will choose to reveal incomplete, and perhaps untruthful, information about its type if that leads to an individually preferable outcome. The central goal in mechanism design is to address this problem of self-interest, and design incentives to encourage agent behavior that lead to optimal system wide solutions to distributed multi-agent optimization problems. Formally the problem of CMD can be defined as below. Following is a list of symbols which will be used throughout the paper.

$\mathcal{I}$	$=$	Set of agents in the system, indexed by $i = 1, \ldots, I$
$\mathcal{O}$	$=$	Set of outcomes
o	$=$	A particular outcome
θ_i	$=$	Type (preference) of agent i. It determines agent's utility over different outcomes
Θ_i	$=$	All possible preferences available to agent i
θ	$=$	$\theta_1 \times \theta_2 \times \ldots \times \theta_I$
Θ	$=$	$\Theta_1 \times \Theta_2 \times \ldots \times \Theta_I$
$u_i(o, \theta_i)$	$=$	Utility of agent i for outcome o when his type is θ_i. This is quantitative measure of agent's happiness for the outcome

$$f : (\Theta) \mapsto \mathcal{O} = \text{Social choice function}$$

The system wide goal in a CMD is to implement social choice function $f(\theta)$. On the surface, the problem looks as simple as just asking the agents to report their types and then computing the solution to the function $f(\theta)$. Unfortunately, the life is not so simple. The agents are self-interested and rational so there is no reason they must report their true type.

Consider for example a network routing problem in which the system-wide goal is to allocate resources to minimize the total cost of delay over all agents, but each agent has private information about parameters such as message size and its unit cost of delay. A typical approach in mechanism design is to provide incentives (for example with suitable payments) to promote truth revelation from agents, such that solution can be computed to the distributed optimization problem. The next section presents a comprehensive description about such approaches, which are typically followed in the design of classical economic mechanisms.

III. Classical Theory of Mechanism Design

Mechanism design is the sub-field of microeconomics and game theory that considers how to implement good system wide solution to problems that involve multiple self-interested agents, each with private information about their preferences. This section provides an introduction to the game-theoretic approach to mechanism design.

A. A Brief Introduction to Game Theory

Game theory [8] is a method to study a system of self-interested agents in the condition of strategic interaction.

1) Basic Definitions:

(i)Type of an Agent: This concept has already been introduced in Section II.

(ii) Strategy: A strategy is a decision rule, that defines the action an agent will select in every distinguished state of world. Let Σ_i denotes the set of all possible strategies available to an agent i. The strategy which agents i selects depends on its type θ_i, and hence the strategy selected by the agent is typically represented by a mapping $s_i : \Theta_i \mapsto \Sigma_i$ and we denote the strategy selected by agents i as $s(\theta_i)$. Sometimes when it is clear from the context, the conditioning on agent's type is left implicit, and we write s_i for the strategy selected by agent i.

(iii) Outcome Rule: Outcome rule, $g(.)$, is a mapping from agents strategies to the set of all possible outcomes i.e. $g : \Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_I \mapsto \mathcal{O}$.

(iv) Utility in a Game: The utility, $u_i(.)$, of agent i determines its preference over its own strategy and the strategies of other agents, given its type θ_i, which determines its base preferences over different outcomes. Thus $u_i(.)$ is a mapping from $(\mathcal{O} \times \Theta_i)$ to $\Re$. However, we can express it as $u_i : \Sigma_1 \times \Sigma_2 \times \ldots \times \Sigma_I \times \Theta_i \mapsto \Re$ by virtue of the outcome rule.

Thus a *game* defines the set of actions available to an agent and a mapping from agents strategies to an outcome. The basic model of game theory assumes the agents are rational and select a strategy that maximizes its expected utility, given its preference θ_i, beliefs about the strategies of other agents, and structure of the game. Now the obvious question is that given a well defined game, is there a way one can analyze the game theoretically and findout which strategy each agents is going to play. In game theory this question is same as asking does there exist a solution of the game and if it does then the corresponding strategies of the agents will be called as equilibrium strategies. The Game theory provides a number of solution concepts to compute the outcome of a game and equilibrium strategies of the agents. The next section delineate three solution concepts useful for rest of the paper.

2) Solutions Concepts: It is useful to introduce following notation before we start describing various solution concepts.

$$\begin{aligned}
s(\theta) &= (s_1(\theta_1), s_2(\theta_2), \ldots, s_I(\theta_I)) \\
s_{-i}(\theta) &= (s_1(\theta_1), \ldots, s_{i-1}(\theta_{i-1}), s_{i+1}(\theta_{i+1}), \ldots, s_I(\theta_I)) \\
s(.) &= (s_1(.), s_2(.), \ldots, s_I(.)) \\
s_{-i}(.) &= (s_1(.), \ldots, s_{i-1}(.), s_{i+1}(.), \ldots, s_I(.)) \\
\Sigma &= (\Sigma_1, \Sigma_2, \ldots, \Sigma_I) \\
\Sigma_{-i} &= (\Sigma_1, \ldots, \Sigma_{i-1}, \Sigma_{i+1}, \ldots, \Sigma_I) \\
\theta_{-i} &= (\theta_1, \ldots, \theta_{i-1}, \theta_{i+1}, \ldots, \theta_I)
\end{aligned}$$

$$\begin{aligned}
u_i\left(s_i(\theta), s_{-i}(\theta_{-i}), \theta_i\right) &= \text{Utility of agent } i \text{ for agents types} \\
&\quad\; \theta \text{ and strategy profile } s(\theta) \\
F(\theta) &= \text{Distribution function for agents type} \\
u_i\left(s_i(\theta), s_{-i}(.), \theta_i\right) &= \text{Expected Utility of agent } i \text{ over } F(.) \\
&\quad\; \text{if his type is } \theta_i \text{ and rest of the agents} \\
&\quad\; \text{follow the strategy } s_{-i}(.)
\end{aligned}$$

(i). Nash Equilibrium: It says that under the assumption of common knowledge (i.e. each agent's type and strategy set is known to every other agent in the game) a strategy profile $s = (s_1, s_2, \ldots, s_I)$ is an equilibrium profile, called Nash equilibrium, if for every i,

$$u_i\left(s_i(\theta_i), s_{-i}(\theta_{-i}), \theta_i\right) \geq u_i(s_i^{'}(\theta_i), s_{-i}(\theta_{-i}), \theta_i) \forall s_i^{'} \neq s_i$$

Although Nash equilibrium is fundamental into game theory, it makes very strong assumption about common knowledge which makes it useless for computational mechanism design.

(ii). Dominant Strategy Equilibrium: It is a stronger solution concept which says that a strategy profile $s = (s_1, s_2, \ldots, s_I)$ is said to be in dominant strategy equilibrium if for every i,

$$\begin{aligned}
u_i\left(s_i(\theta_i), s_{-i}(.), \theta_i\right) &\geq u_i(s_i^{'}(\theta_i), s_{-i}(.), \theta_i) \\
&\quad \forall s_i^{'} \neq s_i, s_{-1} \in \Sigma_{-i}
\end{aligned}$$

Dominant-strategy equilibrium is very robust solution concept, because it makes no assumptions about the information available to agents about each other, and doesnot require an agent to believe that other agents will behave rationally to select its own optimal strategy.

(iii). Bayesian-Nash Equilibrium: A strategy profile $s = (s_1, s_2, \ldots, s_I)$ is said to be in Bayesian-Nash equilibrium if for every i,

$$\begin{aligned}
u_i\left(s_i(\theta_i), s_{-i}(.), \theta_i\right) &\geq u_i(s_i^{'}(\theta_i), s_{-i}(.), \theta_i) \\
&\quad \forall s_i^{'} \neq s_i
\end{aligned}$$

The key difference in Bayesian-Nash and Nash equilibrium is that in Bayesian-Nash agent i's strategy $s_i(\theta_i)$ is the best response to other agents' strategies in the expected sense over the type of the agents. On the other hand, in dominant strategy equilibrium, agent i's strategy is best response to any choice of other agents' strategies in the expected sense over the type of the agents.

B. Mechanism Design:Important Concepts

Whether it is computational mechanism design or classic economic mechanism design, the central problem is to implement an optimal system-wide solution to a decentralized optimization problem. The system-wide goal in mechanism design is defined with a *social choice function*. The following are exact definitions of social choice function, mechanism, mechanism implementation, and mechanism design.

(i). Social Choice Function: A mapping $f : \Theta_1 \times \Theta_2 \times \ldots \times \Theta_I \mapsto \mathcal{O}$ is known as social choice function.

(ii) Mechanism: A mechanism $\mathcal{M} = (\Sigma_1, \Sigma_2, \ldots, \Sigma_I, g(.))$ defines the set of strategies Σ_i available to each agent, and an *outcome rule* $g : \Sigma_1, \Sigma_2, \ldots, \Sigma_I \mapsto \mathcal{O}$, such that $g(s)$ is the outcome implemented by the mechanism for the strategy profile $s = (s_1, \ldots, s_I)$.

(iii) Mechanism Implementation: Mechanism $\mathcal{M} = (\Sigma_1, \Sigma_2, \ldots, \Sigma_I, g(.))$ is said to implement social choice function $f(\theta)$ if $g\left(s_1^*(\theta_1), \ldots, s_I^*(\theta_I)\right) = f(\theta)$, for all $\theta \in \Theta$, where strategy profile $\left(s_1^*(\theta_1), \ldots, s_I^*(\theta_I)\right)$ is an equilibrium solution to the game induced by $\mathcal{M}$.

(iv) Mechanism Design: The mechanism design problem is to design a mechanism- a set of possible agent strategies and an outcome rule-to implement a social choice function with desirable properties, in as strong a solution concept as possible, i.e. dominant is preferred to Bayesian-Nash because it makes less assumptions about agents.

C. Properties of Social Choice Functions

Many properties of a mechanism are stated in terms of the properties of the social choice function that the mechanism implements. This section describes three important properties of social choice function: (1)Allocative Efficiency, (2)Budget Balance, (3) Weak Budget Balance. All the three properties are based on the assumption that agents have quasi-linear utility functions and the underlying mechanism is quasi-liner mechanism.

(i)Quasi-Linear Utility: An agent i is said to have quasi-linear utility if the utility function is of the form:

$$u_i(o, \theta) = v_i(x, \theta_i) - p_i$$

where it is assumed that outcome o defines a *choice* $x \in \mathcal{K}$ from a discrete set and a *payment or transfer* p_i by the agent. The function $v_i(x, \theta)$ is known as *valuation function* for agent i. In an allocation problem, for example, with quasi linear utility function of the agents; the alternatives $\mathcal{K}$ represent allocations, and the transfer represents payments to the auctioneer.

With quasi-linear agent preferences, the outcome of the social choice function can be expressed by collective outcome of the choice $x(\theta)$ and payment $p_i(\theta)$ for each agent i:

$$f(\theta) = (x(\theta), p_1(\theta), \ldots, p_I(\theta))$$

Similarly, the outcome rule, $g(s)$, can also be decomposed into a choice rule, $k(s)$, that selects a choice from the choice set given strategy profile s, and a transfer rule $t_i(s)$ that selects a payment for agent i based on strategy profile s.

(ii)Quasi-Linear Mechanism: A quasi-linear mechanism $\mathcal{M} = (\Sigma_1, \ldots, \Sigma_I, k(.), t_1(.), \ldots, t_I(.))$ defines: the set of *strategies* Σ_i available to each agent; a *choice rule* $k : \Sigma \mapsto \mathcal{K}$, and *transfer rules* $t_i : \Sigma \mapsto \Re$, one for each agent i.

(iii)Allocative Efficiency: Social choice function $f(\theta) = (x(\theta), p(\theta))$ is *allocative-efficient* if

$$\sum_{i=1}^{I} v_i(x(\theta), \theta_i) \quad \geq \quad \sum_{i} v_i(x^{'}(\theta), \theta_i)$$
$$\forall\, x^{'}(.) : \Theta \mapsto \mathcal{K};\ \theta \in \Theta$$

It is common to state this as *allocative efficiency*, because the choice sets often define an allocation of items to agents. An efficient allocation maximizes the total value over all agents.

(iv)Budget Balance: Social choice function $f(\theta) = (x(\theta), p(\theta))$ is *budget balance* if

$$\sum_{i=1}^{I} p_i(\theta) = 0\ \forall\, \theta \in \Theta$$

In other words, there are no net transfers out of the system or into the system.

(v)Weak Budget Balance: Social choice function $f(\theta) = (x(\theta), p(\theta))$ is *weakly budget-balanced* if

$$\sum_{i=1}^{I} p_i(\theta) \geq 0\ \forall\, \theta \in \Theta$$

In other words, there can be net payment made from agents to the mechanism, but no net payment from the mechanism to the agents.

D. Properties of Mechanisms

The definition follow quite naturally from the concept of implementation and properties of social choice functions. A mechanism has property P if it implements a social choice function with property P. This means the mechanism is said to be allocative efficient if it implements an allocative efficient social choice function. The budget balanced and weakly budget-balanced mechanism can also be defined in the same way. An important property of the mechanism is still in order: *Individual Rationality*.

(i)Individual Rational Mechanism: A mechanism $\mathcal{M}$ is individual-rational if for all preferences θ_i it implements a social choice function $f(\theta)$ with

$$u_i\left(f(\theta_i, \theta_{-i})\right) \geq \overline{u_i}\left(\theta_i\right)$$

where $u_i\left(f(\theta_i, \theta_{-i})\right)$ is the *expected utility* for agent i at the outcome, given distributional information about the types of other agents, $F(\theta)$, and $\overline{u_i}\left(\theta_i\right)$ is the expected utility for non-participation. In other words, a mechanism is individual-rational if an agent can always achieve as much expected utility from participation as without participation, given prior beliefs about the preferences of other agents.

E. Incentive Compatible, Direct-Revelation Mechanism

In this section we define an important class of mechanism called as incentive compatible direct revelation mechanism. These mechanism are the central idea behind the powerful theoretical tool in mechanism design: *Revelation Principle* which is the topic of next section and the prime motivating factor for this paper.

(i)Direct-Revelation (DR) Mechanism: A mechanism $\mathcal{M}$ is known as direct revelation if the strategy of each agents is to reveal its own type (not necessarily true type). In otherwords, $\Sigma_i = \Theta_i \; \forall i \in \mathcal{I}$.

(ii)Truth Revealing Strategy: A strategy $s_i \in \Sigma_i$ is truth-revealing if $s_i(\theta_i) = \theta_i \; \forall \theta_i \in \Theta_i$

(iii)Incentive Compatible(IC) DR Mechanism: A direct revelation mechanism is called incentive compatible mechanism if the truth revealing strategy $s^* = (s_1^*, \ldots, s_I^*)$ is solution to the game induced by mechanism.

Depending upon the type of equilibrium the truth-revealing strategy has, the IC-DR mechanisms can be further divided into two categories:

(A)Bayesian-Nash IC-DR Mechanism: A DR mechanism $\mathcal{M}$ is Bayesian-Nash incentive-compatible if truth-revelation is a Bayesian-Nash equilibrium of the game induced by the mechanism.

(B)Dominant-Strategy IC-DR Mechanism: A DR mechanism $\mathcal{M}$ is dominant-strategy incentive-compatible (also called *strategy-proof*) if truth-revelation is a dominant-strategy equilibrium of the game induced by the mechanism.

F. The Revelation Principle

The revelation principle states that under quite weak conditions any mechanism can be transformed into an equivalent IC-DR mechanism that implements the same social choice function. The revelation principle is summarized in two theorem below.

(i)Dominant-Strategy Revelation Principle: Suppose there exists a mechanism (direct or otherwise) $\mathcal{M}$ that implements the social-choice function $f(.)$ in dominant strategy equilibrium. Then $f(.)$ is truthfully implementable in dominant strategies IC-DR, i.e. in a strategy-proof mechanism.

(ii)Bayesian-Nash Revelation Principle: Suppose there exists a mechanism (direct or otherwise) $\mathcal{M}$ that implements the social-choice function $f(.)$ in Bayesian-Nash equilibrium. Then $f(.)$ is truthfully implementable in a Bayesian-Nash IC-DR mechanism.

One interpretation of the revelation principle is that incentive-compatibility comes for free. In other words, if an indirect-revelation and/or non-truthful mechanism solves a distributed optimization problem, then we would also expect a direct-revelation truthful implementation. The revelation principle states what can be achieved, what cannot be achieved, but without stating the *computational structure* to achieve a particular set of properties. For example:

- Suppose $\mathcal{M}^{'}$ is the class of the direct mechanism with useful properties $P1, P2$, and $P3$. It follows that any mechanism m with these three properties must be "outcome equivalent" to a direct mechanism in $\mathcal{M}^{'}$, in the sense that m must implement same outcome as a mechanism in this class for all possible agent types.
- Suppose that *no* direct mechanism had properties $P1, P2$, and $P3$. It follows that there can be no mechanism (direct or otherwise) with properties $P1, P2$, and $P3$.

G. Vickrey-Clarke-Groves Mechanisms

Vickrey-Clarke-Groves mechanisms, often simply called Groves mechanisms, are the only *dominant-strategy IC-DR* mechanisms for the quasi linear preferences which are *allocative efficient* also. The *Pivotal* mechanism is a special type of VCG mechanism in which payment rule is carefully set to achieve an additional property: *individual-rationality*. The pivotal mechanism is called as *Generalized Vickrey Auction* mechanism when it is applied to *Combinatorial Allocation Problem (CAP)*.

In this section we first briefly summarize the VCG mechanism without providing proof for its efficiency, strategy proofness, and uniqueness. Next, we define Pivotal mechanisms.

VCG Mechanism: Consider a DR mechanism with quasi-linear utility for all agents. As discussed in the previous section, for such kind of mechanisms, we write the outcome rule $g\left(\hat{\theta}\right)$ in terms of *choice rule*, $k : \Theta \mapsto \mathcal{K}$, and a *payment rule*, $t_i : \Theta \mapsto \Re$, for each agent i.

In Groves mechanism, agent i reports type $\hat{\theta}_i = s_i(\theta_i)$, which may not be its true type. Given reported types $\hat{\theta} = \left(\hat{\theta}_1, \ldots, \hat{\theta}_I\right)$, the choice rule in a Groves mechanism computes:

$$k^*\left(\hat{\theta}\right) = \arg\max_{k \in \mathcal{K}} \sum_i v_i(k, \hat{\theta}_i) \quad (1)$$

Choice k^* is the selection that maximizes that total reported value over all agents.

The payment rule in a Groves mechanism is defined as:

$$t_i\left(\hat{\theta}\right) = h_i\left(\hat{\theta}_{-i}\right) - \sum_{j \neq i} v_j\left(k^*(\hat{\theta}), \hat{\theta}_j\right) \quad (2)$$

where $h_i : \Theta_{-i} \mapsto \Re$ is an arbitrary function on the types of every agent except i. This can be proved that any Quasi-linear,DR mechanism in which the choice and payments are evaluated by equation 1 and 2 respectively will be allocative efficient in addition to strategy proofness. The freedom in selecting functions $h_i(.)$ in VCG mechanisms leads to the description of a "family" of mechanisms. Different choices makes different tradeoffs across budget-balance and individual-rationality. One such family is Pivotal mechanism.

Pivotal Mechanism: The Pivotal mechanism computes the additional transfer term $h_i(.)$ as:

$$h_i\left(\hat{\theta}_{-i}\right) = \sum_{j \neq i} v_j\left(k^*_{-i}(\hat{\theta}_{-i}), \hat{\theta}_j\right) \quad (3)$$

where $k^*_{-i}(\hat{\theta}_{-i})$ is the *optimal collective choice* for with agent i taken out of the system:

$$k^*_{-i}(\hat{\theta}_{-i}) = \arg\max_{k \in \mathcal{K}} \sum_{j \neq i} v_j(k, \hat{\theta}_j) \quad (4)$$

It can be proved that Pivotal mechanism are individual rational under very mild conditions.

In next section, we first describe the CAP and then show that how Pivotal mechanism can solve the CAP.

H. CAP and GVA

When Pivotal mechanism is applied to the combinatorial allocation problem it takes the form of traditional sealed-bid combinatorial auction, often called the *Generalized Vickrey Auction* (GVA). In this section first we define CAP to motive the GVA and then show how GVA solves the CAP.

1) Combinatorial Allocation Problem (CAP) : The combinatorial allocation problem (CAP) is a resource allocation problem in which a set of items are to be allocated across a set of agents. Agents are assumed to have non-linear values for bundles of items, and the goal is to determine the allocation that maximizes the total value over all agents.

The CAP is relevant to many interesting and important real-world applications, including scheduling, logistics and network computation domains. Indeed, it has attracted considerable recent attention in the academic literature because of its application to the FCC spectrum. Following are two compelling examples of the situations where the problem can be formulated as CAP.

(i)Travel Packages: Consider the allocation of flights, hotel rooms, and entertainment tickets to agents that represent clients with different preferences over location, price, hotels, and entertainment. Moreover, a client has no value for an outward flight without a matching return flight or a hotel room without a flight.

(ii)Airport landing and takeoff scheduling: A number of take-off and landing slots are available across airports in the U.S. Competing airlines need pairs of takeoff and landing slots that are compatible with flight-time and schedule requirements.

The CAP can be framed as an optimization problem. Let

$$\begin{aligned}
\mathcal{G} &= \text{A set of discrete items to be allocated to agents} \\
|\mathcal{G}| &= G \\
\mathcal{I} &= \text{A set of agents} \\
|\mathcal{I}| &= I \\
v_i &: 2^G \mapsto \Re_+ = \text{Valuation function for agent } i \\
S &= \text{A subset of } \mathcal{G} \\
\mathcal{K} &= \{(S_1,\dots,S_I) : S_i \cap S_j = \Phi, S_i \subseteq \mathcal{G}\} \\
&= \text{A feasible allocation of items to agents with} \\
&\quad\ \text{agent } i \text{ receiving bundle } S_i
\end{aligned}$$

We make the following assumptions

$$\begin{aligned}
v_i(S) &\geq 0\ \forall S \subset \mathcal{G} \\
v_i(\Phi) &= 0 \\
v_i(S) &\geq v_i(S')\ \forall S \subset S' \text{ i.e. free disposal of items} \\
&\quad\ \text{is allowed}
\end{aligned}$$

Under these assumptions the CAP can be expressed as following optimization problem

$$\text{Max} \sum_{i \in \mathcal{I}} v_i\left(S_i\right) \tag{5}$$

$$\text{s.t.}\ (S_1,\dots,S_I) \in \mathcal{K} \tag{6}$$

An interesting interpretation of the CAP is sealed bid combinatorial auction problem which is discussed in the next section.

2) Sealed Bid Combinatorial Auction (GVA): Consider a situation where a seller (or auctioneer) is willing to sell a set of distinct goods, $\mathcal{G}$. Let there are I potential buyers (or bidders) interested in buying the subset of goods. Each bidder has its own valuation for each subset of the goods. Let the function $v_i(S)$ express the valuation of agent i for subset $S \subseteq \mathcal{G}$. Assume that buyers have positive value for each nonempty subset of goods and zero value for empty subset of goods. Also assume free disposal i.e. $v_i(S) \geq v_i(S')\ \forall S \subset S'$.

Now imagine that auctioneer conducts a sealed bid auction where each bidder submits a (possibly truthful) bid (i.e. a valuation function, $\hat{v}_i(S)$) to the auctioneer. The auctioneer computes the allocation of subsets of goods to the bidder, $S^* = (S_1^*,\dots,S_I^*)$ in his own selfish way and also fixes the prices p_i for each bidder which they have to pay to the auctioneer.

If the bidders are viewed as agents with: actual valuation function, $v_i(.)$, as their true type θ_i and bid, $\hat{v}_i(.)$, as their reported type $\hat{\theta}_i$; then the sealed bid auction will turn out to be a DR mechanism. If prices which bidders pay to the auctioneer are viewed as payments then the difference, $(v_i(S) - p_i)$, can be considered as utility of the agent i and the mechanism will turn into Quasi-Linear Mechanism. For such Quasi-Liner DR mechanism, the set of feasible allocations, $\{S = (S_1,\dots,S_I) : S_i \cap S_j = \Phi, S_i \subseteq \mathcal{G}\}$, becomes the choice set $\mathcal{K}$ and the prices p_i becomes payments. Thus we see that sealed bid combinatorial auction can be put into the framework of quasi-linear DR mechanism. Further, if it is assumed that the choice and transfer rules are same as Pivotal mechanism then each bidder will reveal its true type i.e. $\hat{v}_i(.) = v_i()$ and the allocation $k^*(\hat{v})$ will be an efficient allocation, given by:

$$k^*(\hat{v}) = \arg\max_{k \in \mathcal{K}} \sum_i v_i(S) \tag{7}$$

The payments for agents will be:

$$p_i = (V_{-i}^* - V^*) \tag{8}$$

where V^* is the total value of the best allocation with agent i and V_{-i}^* is the total value of the best allocation without agent i. Such an Pivotal mechanism is known as Generalized Vickrey Auction. Comparing equation (5) and (7) shows that GVA allocation is the solution of the underlying CAP problem.

IV. Computational Concerns in Mechanism Design

The classical mechanism design literature largely ignores computational considerations. It is common to assume that agents can reveal their complete preferences over all possible outcomes (the *revelation principle*), and that the mechanism can solve an optimization problem to select the best outcome (e.g. the *Groves mechanisms*). Much of the classic mechanism design is driven by the revelation principle, however, it is no more useful when we come to the domain of computational mechanism design. The transformation assumed in the revelation principle from indirect mechanism (e.g. an iterative auction) to direct revelation mechanism (e.g. a sealed-bid auction) assumes unlimited computational resources, both for agents in submitting valuation functions, and for auctioneer in computing the outcome of a mechanism. In particular, the revelation principle assumes:

- Agents can compute and communicate their complete preferences
- The mechanism can compute the correct outcome with complete information about all relevant decentralized information in the system.

It can soon become impractical for an agent to compute and communicate its complete preferences to the mechanism, and for mechanism to compute a solution to the centralized optimization problem.

The computation required in a classical mechanism can be characterized within two levels:

- At the agent level:
  - *Valuation complexity.* How much computation is required for an agent to provide information about its preference to the mechanism ?
  - *Strategic complexity.* How much computation an agent needs to model the strategy of other agents and then solve a game-theoretic problem to compute its own strategy.
- At the infrastructure level:
  - *Winner-determination complexity.* How much computation is expected for the mechanism to compute an outcome given information provided by agents?
  - *Communication complexity.* How much communication is required, between agents and mechanism, to compute an outcome?

Dominant strategy mechanisms have excellent strategic complexity as an agent can compute a dominant-strategy without modeling the other agents and without game-theoretic reasoning. However, the direct revelation property of strategy-proof mechanisms, e.g. Groves Mechanisms, provide very bad agent valuation complexity. An optimal bidding strategy requires that an agent determines its complete preference over all possible outcomes.

As mentioned earlier we are interested in CAP which is a known NP-Hard problem. Also, among classical mechanisms GVA are the best known mechanism which solve the CAP. Therefore, naturally we are interested in exploring computational aspects of the GVA. The next section presents a computational perspective of the GVA and also list the challenges while implementing it to solve the CAP.

A. Computation and the GVA

From computational perspective the GVA presents a number of challenges:

- *Winner determination is NP-hard.* Winner determination in the GVA is NP-hard, equivalent to the maximum weighted set packing problem. The auctioneer must solve the winner determination problem once with all agents, and then once more with each agent removed from the system to compute payments, resulting in a total of $(I + 1)$ NP-Hard problems.
- *Agents must* **compute** *values for an exponential number of bundles of items.* The GVA requires complete information revelation from each agent. The valuation problem for a single bundle can be hard, and in combinatorial domains there are an exponential number of bundles to consider.
- *Agents must* **communicate** *values for an exponential number of bundles of items.* Once an agent has determined its preferences for all possible outcomes it must communicate that information to the auctioneer.

A number of proposal exist to address each one of these problems. The first problem, concerning the computational complexity of the auctioneer's winner determination problem, has received most attention. In comparison, the second problem, concerning the complexity on participants to determine their preferences has received considerably less attention. exceptions include the brief discussion of bidding programs in Nisan [4], and the recent progress that has been made on dynamic mechanism [1].

This dynamic approach includes Parkes COMBAUCTION algorithm- an iterative way of computing the allocations without complete information revelation from agents. He also extends the algorithm to VICKAUCTION for computing the vickrey payments of the agents. Iterative method provide interactive solution to problem, requiring just enough information from agents to compute the efficient allocation and vickrey payments.

The Parkes algorithm is an extension to the Bretsekas' primal-dual algorithm [6] for the *assignment problem*- special case of CAP. At the core, COMBAUCTION algorithm computes the solution to the LP formulation of the CAP, introduced by Bickchandani & Ostroy [7], by making use of primal-dual algorithm.

Thus we see that one decade after the Bertsekas work, Parke has again exploited the primal-dual concept of linear programming in an innovative way to solve the complex valuation problem for agents in GVA. This shows the immense power of duality concept for solving the seemingly hard computational problems. Undoubtedly, Parkes work is one of the remarkable contribution in the domain of computational mechanism design. This motivates us to look for synergy between duality concept and computational problems in the domain of mechanism design. We believe that Parkes idea can provide a good starting point in this direction. With this idea in our mind, in Section V, we provide a premier on linear programming theory and concept of duality in next section. It serves as fundamental background needed to understand the Parkes algorithm. Section VI first describes the hierarchy of linear programs for the CAP proposed by Bickchandani & Ostroy and then finally COMBAUCTION.

V. Linear Programming Duality

Associated with any linear programming problem, and intimately related to it, is a corresponding dual linear programming problem. The variables of the dual problem can be interpreted as prices associated with the constraints of the original (primal) problem. In this section we define the dual program that is associated with a given linear program and the duality theorems.

A. Dual Linear Program

Symmetric Form:

Primal		*Dual*	
minimize	cx	maximize	λb
subject to	$Ax \geq b$	subject to	$\lambda A \leq b$
	$x \geq 0$		$\lambda \geq 0$

Unsymmetric Form:

Primal		*Dual*	
minimize	cx	maximize	λb
subject to	$Ax = b$	subject to	$\lambda A \leq b$
	$x \geq 0$		

If A is $m \times n$ matrix, then x is an $n-$dimensional column vector, b is an $m-$dimensional column vector, c is an $n-$dimensional row vector, and λ is an $m-$dimensional row vector. The vector x is the variable of the primal problem, and λ is the variable of the dual problem. It is important to note that role of primal and dual can be reversed.

The variables of the dual problem can be interpreted as prices associated with the constraints of the original (primal) problem, and through this association it is possible to give an economically meaningful characterization to the dual variables whenever there is such a characterization for the primal [9].

For example, if it is the problem a dietician faces while selecting a combination of foods to meet certain nutritional requirements at the minimum cost, then the dual of it will be the problem faced by pharmaceutical company that produces in pill form each of the nutrients considered important by dietician. The pharmaceutical company tries to convince the dietician to buy pills, and thereby supply the nutrients directly rather than through purchase of various foods. The problem faced by drug company is that of determining positive unit prices for the nutrients pills so as to maximize revenue while at the same time being competitive with real food.

To this point the relation between the primal and dual problems has been simply a formal one based on what might appear as an arbitrary definition. The deeper connection between a problem and its dual is expressed by two duality theorems: (i) weak duality (ii) strong duality.

Let $V_P(x) = cx$ is the value of feasible primal solution x, and $V_D(\lambda) = \lambda b$, the value of feasible dual solution λ.

B. Weak Duality

Given a feasible primal solution x with value $V_P(x)$ and a feasible dual solution λ with value $V_D(\lambda)$, then $V_D(\lambda) \leq V_P(x)$.

C. Strong Duality

Primal solution x^* and dual solution λ^* are a pair of optimal solutions for the primal and dual respectively, iff x^* and λ^* are feasible and $V_D(\lambda^*) = V_P(x^*)$.

D. Complementary-slackness

The strong-duality theorem of linear programming can be restated in terms of *complementary-slackness* (CS for short) conditions.

Theorem 1: (CS-unsymmetric form) Let x and λ be feasible solutions for the primal and dual problems, respectively, in unsymmetric form. A necessary and sufficient condition that they both be optimal solution is that for all i

1) $x_i > 0 \Rightarrow \lambda a_i = c_i$
2) $x_i = 0 \Leftarrow \lambda a_i < c_i$

Theorem 2: (CS-unsymmetric form)Let x and λ be feasible solutions for the primal and dual problems, respectively, in symmetric form. A necessary and sufficient condition that they both be optimal solution is that for all i

1) $x_i > 0 \Rightarrow \lambda a_i = c_i$
2) $x_i = 0 \Leftarrow \lambda a_i < c_i$
3) $\lambda_j > 0 \Rightarrow a^j x = b^j$
4) $\lambda_j = 0 \Leftarrow a^j x > b_j$

where a_i and a^j are the $i-$th column and $j-$th row respectively of the matrix A and x_i and λ_i are the $i-$the components of the vector x and λ.

E. Primal Dual Algorithms

A primal-dual algorithm searches for feasible primal and dual solutions that satisfy complementary-slackness conditions, instead of searching for an optimal primal (or dual) solution directly. A standard primal-dual formulation maintains a feasible dual solution, say λ, and computes a solution to a *restricted primal problem*, given the dual solution. The restricted primal is formulated to compute a primal solution that is both feasible and satisfies CS conditions with the dual solution. In general this is not possible (until the dual solution is optimal), and a relaxed solution is computed. The restricted primal problem is typically formulated to compute this relaxed solution in one of two ways:

- Compute a feasible primal solution $x^{'}$ that minimizes the "violation" of complementary-slackness conditions with dual solution λ.
- Compute a primal solution $x^{'}$ that satisfies CS conditions with dual solution λ, and minimizes the "violation" of feasibility conditions.

First method is more useful in the context of iterative auction design because it maintains feasible primal solution, which becomes the provisional allocation in the auction.

VI. COMBAUCTION: A PRIMAL-DUAL ALGORITHM FOR CAP

A. Bickchandani & Ostroy LP Formulation for CAP

Primal-dual based auction methods require linear programming formulation of allocation problems. Bickchandani & Ostroy have formulated a hierarchy of linear programs for the

CAP, introducing additional constraints to remove fractional solutions. Although it is always possible to add enough constraints to a linear program relaxation to make the optimal solution integral, the particular formulation proposed by Bickchandani & Ostroy are interesting because the constraints have natural interpretations as prices in the dual. We describe this hierarchy of LP formulation starting with integer program for CAP.

1) Integer Program Formulation: Problem CAP formulated in Section III-H.1 is an instance of what is known as Set Packing Problem (SPP)- an integer program (IP)- which is described below.

$$\max_{x_i(S)} \sum_S \sum_i x_i(S) v_i(S) \tag{9}$$

$$\text{s.t.} \sum_S x_i(S) \leq 1,\ \forall i \tag{10}$$

$$\sum_{S \ni j} \sum_i x_i(S) \leq 1,\ \forall j \tag{11}$$

$$x_i(S) \in \{0,1\},\ \forall i, S \tag{12}$$

where $x_i(S)$ will take values 1 or 0 depending upon whether subset S is allocated to agent i or not. $v_i(S)$ is the valuation of the agent i for subset S. $S \ni j$ indicates a bundle S that contains item j. The objective is to compute the allocation that maximizes value over all agents, without allocating more than one bundle to any agent (10) and without allocating a single item multiple times (11). Let V_I^* denote the value of optimal allocation.

2) First-order LP Formulation: LP_1 is a direct linear relaxation, which replaces the integral constraints $x_i(S) \in \{0,1\}$ with non-negativity constraints, $x_i(S) \geq 0$. The relaxed integer linear program is called as first-order primal LP_1 and the corresponding dual is called as first-order dual DLP_1.

LP_1

$$\max_{x_i(S)} \sum_S \sum_i x_i(S) v_i(S)$$

$$\text{s.t.} \sum_S x_i(S) \leq 1,\ \forall i$$

$$\sum_{S \ni j} \sum_i x_i(S) \leq 1,\ \forall j$$

$$x_i(S) \geq 0,\ \forall i, S$$

DLP_1

$$\min_{p(i), p(j)} \sum_i p(i) + \sum_j p(j)$$

$$\text{s.t.} p(i) + \sum_{j \in S} p(j) \geq v_i(s),\ \forall i, S$$

$$p(i), p(j) \geq 0,\ \forall i, j$$

The above primal-dual formulation has straightforward interpretation in the context of GVA.

Primal Problem: The primal GVA problem is to allocate items to agents to maximize the sum of values over all agents, such that no item is allocated to more than one agent.

Dual Problem: The dual GVA problem is to assign linear prices p_j to on items $j \in \mathcal{G}$ (linear because the price for bundle $S \subseteq \mathcal{G}$ is $p(S) = \sum_{j \in S} p(j)$), such that the allocation of the bundles to the agents which maximizes their utility is a feasible allocation and the following sum is minimum (i) each agents' maximum utility (difference between agent's value for a bundle and the price) given the prices; and (ii) the sum of prices of all the goods (this will become auctioneer's maximum revenue).

CS conditions: The four CS conditions, for an unsymmetric form, between a feasible primal solution to an allocation $S\,(S_1, \ldots, S_I)$, and a feasible dual solution to linear prices, $p = (p_1, \ldots, p_G)$, are:

- Agent i receives bundle S_i in the provisional allocation iff the bundle maximizes its utility given the prices, and has non-negative utility.
- The provisional allocation $S = (S_1, \ldots, S_I)$ is the revenue-maximizing allocation given the prices.

3) Second-order LP Formulation: It can be shown by giving counter example that LP_1 doesnot always give integer solution. Therefore, introducing the new constraints to the LP_1 gives LP_2 and DLP_2.

LP_2

$$\max_{x_i(S), y(k)} \sum_S \sum_i x_i(S) v_i(S)$$

$$\text{s.t.} \sum_S x_i(S) \leq 1,\ \forall i$$

$$\sum_i x_i(S) \leq \sum_{k \ni S} y(k),\ \forall S$$

$$\sum_{k \in \mathcal{K}} y(k) \leq 1$$

$$x_i(S), y(k) \geq 0,\ \forall i, S, k$$

DLP_2

$$\min_{p(i), p(S), \pi} \sum_i p(i) + \pi$$

$$\text{s.t.} p(i) + p(S) \geq v_i(s),\ \forall i, S$$

$$\pi - \sum_{s \in k} p(S) \geq 0,\ \forall k \in \mathcal{K}$$

$$p(i), p(S), \pi \geq 0.\ \forall i, S$$

where $\mathcal{K}$ is the collection of all the *partitions* of the set of items $\mathcal{G}$ and $k \in \mathcal{K}$ is one way of partitioning the set $\mathcal{G}$. $k \ni S$ represents all those different partitions of the set $\mathcal{G}$ which contain subset $S \subseteq \mathcal{G}$ as one of the member. Constraints $(\text{LP}_2 - 2)$ and $(\text{LP}_2 - 3)$ replace constraint $(\text{LP}_1 - 2)$, and ensure that no more than one unit of every item is allocated. In the context of GVA, the dual variables $p(S)$ can interpreted as bundle prices, and with substitution $p(i) = \max_S \{v_i(S) - p(S)\}$, i.e. the maximal utility to agent i at prices $p(S)$, and $\pi = \max_{k \in \mathcal{K}} \sum_{S \in \mathcal{K}} p(S)$, i.e. the maximal revenue to the auctioneer at prices $p(S)$.

4) Third-order LP Formulation: For (LP_2) also one can produce the counter examples where the solution is fractional. Therefore, introducing new constraints to the second-order linear LP formulation gives a third-order linear program (LP_3) and its dual (DLP_3). Bickchandani & Ostroy have proved the following important theorem which says that linear program (LP_3) can never produce fractional solutions.

The optimal solution to linear program (LP_3) is always integral, and therefore an optimal solution to CAP, with $V^*_{(LP_3)} = V^*_{(DLP_3)} = V^*_{(I)}$.

$$\begin{aligned}
&\mathrm{LP}_3\\
&\max_{x_i(S), y(k)} \sum_S \sum_i x_i(S) v_i(S)\\
&\text{s.t.} \sum_S x_i(S) \le 1,\ \forall i\\
&x_i(S) \le \sum_{k \ni [i,S]} y(k),\ \forall i, S\\
&\sum_{k \in \mathcal{K}'} y(k) \le 1\\
&x_i(S), y(k) \ge 0,\ \forall i, S, k
\end{aligned}$$

$$\begin{aligned}
&\mathrm{DLP}_3\\
&\min_{p(i), p_i(S), \pi} \sum_i p(i) + \pi\\
&\text{s.t.} p(i) + p_i(S) \ge v_i(s),\ \forall i, S\\
&\pi - \sum_{[i,s] \in k} p_i(S) \ge 0,\ \forall k \in \mathcal{K}\\
&p(i), p_i(S), \pi \ge 0.\ \forall i, S
\end{aligned}$$

where partition $\mathcal{K}'$ is an ordered partition over agents which means if $k \in \mathcal{K}'$ then $k = \{[1, S_1], \ldots, [I, S_I]\}$. Variable $y(k)$ will be equal to 0 or 1 depending upon whether partition k is used as an allocation or not. The dual variables $p_i(S)$ are interpreted as *non-anonymous* bundle prices. By non-anonymous, we mean that different prices for the same bundle to different agents. The dual variable $p_i(S)$ is the price to the agent i for bundle S. As before, substitutions $p(i) = \max_S \{v_i(S) - p_i(S)\}$, i.e. the maximal utility to agent i at individual prices $p_i(S)$, and $\pi = \max_{k \in \mathcal{K}'} \sum_{[i,S] \in k} p_i(S)$, i.e. the maximal revenue to the auctioneer at prices $p_i(S)$ given that it can allocate at most one bundle at prices $p_i(S)$ to each agent i.

B. COMBAUCTION: Algorithm

COMBAUCTION is a primal-dual algorithm for the linear program models introduced in the previous section. The algorithm terminates with optimal primal and dual solutions to an appropriate level in the linear-program hierarchy, selecting the price structure dynamically to support the optimal allocation in equilibrium. In COMBAUCTION the prices are linear and anonymous in special cases, non-linear and anonymous (bundle prices) in many problems, and non-linear and non-anonymous when that is necessary to strengthen the dual formulation of the CAP. The decision about anonymous vs. non-anonymous pricing is made dynamically during the algorithm, while non-linear prices are introduced whenever an agent bids for a bundle of items instead of individual items.

Avoiding the technical details of the algorithm we present the central idea behind the algorithm. The important steps of the algorithm are as follows:
(1) Maintain a feasible dual solution (”prices”).
(2) Compute a feasible primal solution (”provisional allocation”) to minimize violations with complementary-slackness conditions given agents’ bids. This is the solution of restricted primal problem. In GVA setting, the restricted primal problem is: Given response bids from each agent allocate bundles to maximize revenue, breaking ties in favour of including more agents in the provisional allocation.
(3)Terminate if all CS conditions are satisfies.
(4)Adjust the dual solution towards an optimal solution, based on CS conditions and the current primal solution (”increase the prices based on agents bids”). The flow diagram of the algorithm is shown in Figure 1. An important observation in connection with above algorithm is still in order. Note that above algorithm computes the allocation of bundles and payments for the agents such that total *reported* value over agents get maximized. Under this scheme of payment is no reason that agents must tell his true value. Therefore, the prices which agents are going to pay at the end of the algorithm may not be Vickrey prices, in otherwords the COMBAUCTION algorithm is not incentive compatible algorithm. To improve upon this Parkes proposed another algorithm VICKAUCTION.

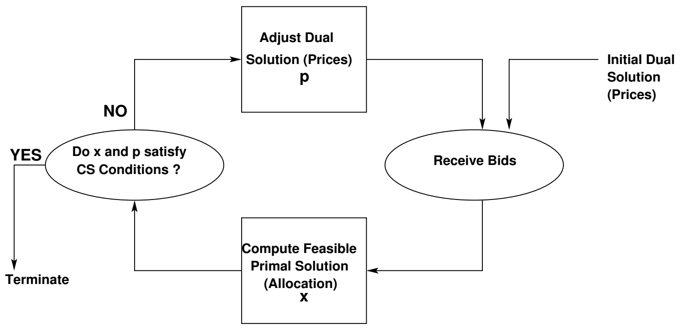

Fig. 1. Flow Chart for COMBAUCTION Algorithm

VII. Conclusions

An important fundamental problem in e-commerce is to design mechanisms that compute optimal system-wide solutions despite the self-interest of individual users (buyers, sellers, brokers, etc.) and computational agents (buying agents, selling agents, etc.). This class of problems, formally called computational mechanism design problems, are similar in nature to the economic mechanism design problems arising in economics and game theory. Classic game-theoretic solutions for these problems are often prohibitively expensive computationally. A plethora of proposals has emerged in recent times to address these problems. One of the popular approaches proposed is based on using the primal-dual algorithm for solving the mechanism design problem.

In this paper, we have brought out the challenges and difficulties in adopting the classical game-theoretic solutions to computational mechanism design problems. We have attempted to bring out the important role duality can play in computational mechanism design. In the last few years, many researchers have explored the use of duality in a wide variety of ways, in designing mechanisms for e-commerce. The examples include: (1)iterative auctions for muti-objects and heterogeneous goods [10], [11], [12], [13], [14];(2) smart markets for industrial procurements [15]; (3) set packing problem for e-commerce applications [7]; (4) shortest path problems[16]; and (5) Parkes recent paper on iterative GVA [17]. Most recent work is due to Bickchandani *et al* [18] where they have up with a LP formulation of CAP whose dual variables are vickrey prices. Hence, unlike the VICKAUCTION, any iterative mechanism which makes use of this formulation needs to solve only one LP and its dual rather than $I + 1$ LPs to compute vickrey payments for agents. A future version of this article will include a comprehensive review of relevant work.

REFERENCES

[1] David C Parkes, *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*, Ph.D. thesis, Computer and Information Science, University of Pennsylvania, 2001.

[2] Andreu Mas-Colell, Michael D Whinston, and Jerry R Green, *Microeconomic Theory*, Oxford University Press, 1995.

[3] Hal R Varian, "Economic mechanism design for computerized agents," in *USENIX Workshop on Electronic Commerce*, 1995.

[4] Noam Nisan, "Bidding and allocation in combinatorial auctions," in *2nd ACM Conf. on Electronic Commerce (EC-00)*, 2000, pp. 1–12.

[5] E H Clarke, "Multipart pricing of public goods," *Public Choice*, vol. 11, pp. 17–33, 1971.

[6] Dimitri P Bertsekas, "The auction algorithm for assignment and other network flow problems: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 133–149, 1990.

[7] Sushil Bikchandani and Joseph M Ostroy, "The package assignment model," Tech. Rep., Anderson Graduate School of Management and Department of Economics, UCLA, 1999.

[8] John Nash, "Equilibrium points in n-person games," in *Natioanl Academy of Sciences*, 1950, vol. 36, pp. 48–49.

[9] David G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley Publishing Company, 1982.

[10] Christine DeMartini, Anthony M. Kwasnica, John O. Ledyard, and David Porter, "A new and improved design for multi-object iterative auctions," 1999.

[11] Lawrence M. Ausubel, "An efficient dynamic auction for heterogeneous commodities," working paper, 2002.

[12] S. Bickchandani and Joseph M. Ostroy, "Ascending price vickrey auction," working paper, 2001.

[13] Sushil Bickchandani, "Auctions of heterogeneous objects," *Games and Economics Behaviour*, vol. 26, pp. 193–220, 1999.

[14] Lawrence M. Ausubel and Paul R. Milgrom, "Ascending auctions with package bidding," *Frontiers of Theoretical Economics*, vol. 1, no. 1, 2002.

[15] Jeremie Gallien and Lawrence M. Wein, "Design and analysis of a smart market for industrial procurement," 2002.

[16] John Hershberger and Subhash Suri, "Vickrey prices and shortest paths: What is an edge worth?," 1999.

[17] David C. Parkes, "An iterative generalized vickrey auction: Strategy-proofness without complete revelation," 2002.

[18] S. Bickchandani, S. de Vries, J. Schumer, and R. Vohra, "Linear programming and vickrey auction," working paper, 2001.