Project: A survey and Critique on Algorithmic Mechanism Design

1 Motivation

One can think of many systems where agents act according to their self interest. Therefore, when we are in the process of designing a system it is always more reasonable to assume that the agents in that system will act as to maximize their self interest alone. In algorithm mechanism design, we are trying to design mechanisms where it is within the best interests of the agents to cooperate. This line of thought started with John Nash who coined the Nash equilibrium to find the best solution for players in non-cooperative games. However, even though a Nash equilibrium is always guaranteed to exist there exists no efficient algorithm to compute it efficiently as it lies in a difficult category of problems Polynomial Parity Arguments on Directed graphs: PPAD complete for 3 or more agents [2]. However, algorithm designers still need to find the best solution to a mechanism problem in an efficient manner. As such algorithmic mechanism design is a curious problem for complexity theorists but also one with immense practical scope. We have to apply algorithmic mechanism design to any system with selfish agents. Below we illustrate some examples for which algorithmic mechanism design is essential:

1.1 Networking

Routers connected to the internet are built by different manufacturers who want to show why their designs and products are superior. As such every router connected to the interest has a selfish interest of transmitting only its own packets and could hypothetically drop everyone else’s packets. However, if every agent was that selfish than data transmission on the internet would be impossible. That is why it might be more appropriate to design a mechanism to reward routers who do not mislead the system about the actual number of packets they are buffering. It is easy to see how algorithmic mechanism design could also be of help in any application where there agents are transmitting to each other information and ultimately only care about their messages being transmitted. (Distributed Systems, Communication Systems...)

1.2 Social Sciences

Truly intelligent agents realize that the long term rewards of cooperation is generally larger than short term rewards of selfishness. (Greedy is rarely optimal). Evolution tells us that agents who were more willing to cooperate were more likely to survive and History tells us that more can be produced by a community than can be produced by a single agent. It is true that evolution and history are local hill climbing algorithms to mechanism design problems but if local solutions have been so powerful in shaping the world than optimal solutions will help us design intelligent agents faster than nature did.

2 What is a Mechanism Design Problem?

A mechanism design problem composed of agents that can take decisions independently and those decisions influence the total output of the mechanism. Specifically for mechanism de-
sign each agent $i$ has a private knowledge of its own type $t^i \in T^i$. The total output $o \in O$ is mapped to a vector of the "reported" types that agents choose to disclose $t = t^1, t^2, \ldots, t^n$ (agents can choose to incorrectly disclose their type if it suits them). Naturally every agent will desire or value outputs more than others we therefore define an agent’s valuations of a certain system output to be $v_i(t^i, o)$. The total utility of an agent $i$ is thus his valuation for the output and the payment $p^i$ he receives from the mechanism $u^i = p^i + v^i(t^i, o)$. More formally we provide the definition of a mechanism.

**Definition (Mechanism)**

1. A mechanism $m$ is defined as $m = (o, p)$ where $o()$ is an output function and $p$ is a vector of size $n$ where $n$ is the total number of agents in our system.

2. The agent can choose a strategy from a family of strategies $a^i \in A^i$ or in this case $t^i \in T^i$. Where an agent’s strategy is to simply report its type, we call this the revelation principle.

3. The mechanism will then output an output function $o(a^1, \ldots, a^n)$

4. For each agent $i$ there exists one strategy $a^i \in A^i$ we call the dominant strategy

3 **Vickrey-Groves-Clarke Mechanisms**

Vickrey-Groves-Clarke (VGC) Mechanisms were first designed to help solve the sealed bid auction problem whereas individuals report bids/valuations for certain items with the winner (person with the highest bid) having to pay the second highest bid to acquire the item. The framework insures that the optimal strategy for every individual is to truthfully report their valuations. VGC mechanisms can further be generalized to any utilitarian system defined by the total good as a weighted or unweighted sum of individual good or more formally:

**Definition** A mechanism maximization problem is called utilitarian if the mechanism output $g(o, t) = \sum_i v_i(t^i, o)$

**Definition** We say that a direct revelatio mechanism $m = (o(t), p(t))$ belongs to the VGC family if

1. $o(t) \in \arg\max_o(\sum_{i=1}^n v_i(t^i, o))$

2. $p^i(t) = \sum_{j \neq i} v_j(t^j, o(t)) + h^i(t^{-i})$ where $h^i()$ is an arbitrary function of $t^{-i}$

We realize that (2) can sometimes be confusing but it can be understood in the following manner. $\sum_{j \neq i} v_j(t^j, o(t))$ simply refers to the value of the output function given that agent $i$ did not exist. $h^i(t^{-i})$ refers to the added value of a response from agent $i$. As such the payment $p^i(t)$ we give agent $i$ is equivalent to how much value his response added to the total output.

**Theorem** (Groves 1973) A VGC mechanism is truthful i.e Telling the truth is the best possible strategy in a VGC mechanism
4 Mechanism Design on the Shortest Path

Nisan and Ronen use the Shortest Path problem in order to demonstrate how mechanism design can be used to solve algorithmic problems. The goal of mechanism design is to create a scenario in which the agents’ best strategy is to be truthful. For the shortest path problem, we consider an edge in a graph as a single agent and our objective is to find the shortest path from x to y. This is only possible when all agents truthfully report their values, after which we can use Dijkstra’s Algorithm to discern the shortest path for the entire graph.

Formal Specification:
The set of feasible outputs O is all paths in the graph from x to y.
An output \( o \in O \) is denoted by a set of agents in the path.
The type \( t \) of agent \( e \) in the type vector \( t \) is the weight of the edge that the agent represents.
The valuation \( v(e, o) \) of agent \( e \) is given by \( -t \) if \( e \in o \). Otherwise \( v(e, o) = 0 \)
The objective function of an output \( o \) is the sum of all valuations of agents in \( o \). Since the valuations of an agent not in \( o \) is 0, \( g(o, t) = \sum v(t, o) \)

we can create a payment function for the Shortest Path problem in such a way that it fits into the VGC mechanism definition. Using the properties of a VGC mechanism, it will thus be the case that the agents will be truthful.

A VGC mechanism requires that:
\( o(t) = \arg\max_o \sum v(t, o) \) and 
\( p(t) = \sum_{j \neq i} v(t, o(t)) + h(t^{-1}) \)

since the goal is to minimize path length, and our objective function \( g(o, t) = \sum v(t, o) \), the best solution is \( \arg\max_o g(o, t) \) since the agent valuations are negative. Thus \( o(t) = \arg\max_o \sum v(t, o) \) is already satisfied.

Therefore we set the payment function as \( p(t) = \sum_{j \neq i} v(t, o(t)) + h(t^{-1}) \) where we let \( h(t^{-1}) \) be negative of the length of shortest path given the agent \( i \) is not part of the solution. A negative value for \( h(t^{-1}) \) is chosen in order to minimize the necessary payments to the agents.

Now that the mechanism is designed to be a VGC mechanism, agents are guaranteed to be truthful. To obtain the shortest path, one simply needs to apply Dijkstra’s Algorithm given the known accurate path values given by each agent.

Time Analysis:
The actions that must be taken to solve the Shortest Path problem is calculating the payment function for every agent, and applying Dijkstra’s Algorithm at the very end.
We break \( p(t) = \sum_{j \neq i} v(t, o(t)) + h(t^{-1}) \) into two parts.
\( \sum_{j \neq i} v(t, o(t)) \) can be calculated in constant time if we calculate \( g(o(t), t) \) before hand as 
\( \sum_{j \neq i} v(t, o(t)) = g(o(t), t) - v(t, o(t)) \) \( g(o(t), t) \) takes time \( n \).
\( h(t^{-1}) \) is the shortest path over the graph which does not include agent \( i \). As such, Dijkstra’s algorithm is called once per agent \( i \). Realistically, if the agent representing an edge is not in the optimal solution, removing that edge from the graph does not change the optimal path. However, the optimal path is still bounded by the number of edges with the worst case being a graph consisting of a single path. Thus it is still bounded by \( n \). Therefore the entire time complexity is \( n + O(Dijkstra’s \ Algorithm) + n^2O(Dijkstra’s \ Algorithm) \) which
is $O(n^2 \log(n))$.

5 Mechanism Design on Network Flow

Nisan and Ronen claim that we can utilize the properties of VGC mechanisms in order to solve other graph problems in a similar way. This section will design a mechanism in order to solve the Network Flow problem using the same principles as the Shortest Path problem. For the graph, we have an agent represent an edge in the graph and our goal is to have each agent truthfully discern its type so that we can run a network flow algorithm on the reported values to find the maximum flow from node $s$ to $d$.

**Formal Specification:**

The type $t^e$ of agent $e$ in the type vector $t$ is the capacity of the edge that the agent represents, and a boolean value of whether it is connected to the terminal node $d$. $t^i = (\text{capacity}_i, \text{adjacentToDestination}_i)$

The set of feasible outputs $O$ is a set of every agent, and the flow that passes through it $\forall_i (\text{agent}_i, \text{flow}_i) s.t. \text{flow}_i \leq \text{capacity}_i$.

The valuation $v^e(t^e, o)$ of agent $e$ is given by $\text{flow}_e$ if adjacentToDestination$_i = true$. Otherwise, we can let $v^e(t^e, o) = 0$. Additionally, we can let $v^e(t^e, o) = -\infty$ if ever $\text{flow}_i \geq \text{capacity}_i$ as this creates an invalid output.

The objective function of an output $o$ is the sum of all valuations of agents in $o$. Since the valuations of an agent not adjacent to $d$ is 0, $g(o, t) = \sum_i v^i(t^i, o)$ which is equal to the flow into destination node $d$. The goal of our algorithm is to obtain the $o(t)$ that maximizes $g(o, t)$.

We define the payment function as $p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) + h^i(t^{-1})$ where we define $h^i(t^{-1})$ to be the maximum flow given the agent $i$ is not part of the solution.

We see that this mechanism conforms to VGC as the VGC mechanism requires that $o(t) = \text{argmax}_o \sum_i v^i(t^i, o)$ and $p^i(t) = \sum_{j \neq i} v^j(t^j, o(t)) + h^i(t^{-1})$ of which both fulfilled. Therefore we can guarantee that the agents tell the truth.

**Time Analysis:**

The time taken for this problem is performing Network Flow using the reported edge values, and calculating the payment function for each agent. The payment function time complexity must compute the Network Flow subproblem for each agent, under the assumption that agent is not part of the graph plus a constant time factor to calculate $\sum_{j \neq i} v^i(t^j, o(t))$. The Network Flow Algorithm’s time complexity is dependent on the relation of nodes to edges in the graph so we will not make any assumptions on the time complexity of Network Flow. Thus the total time complexity is $n + O(\text{Network Flow}) + n^*O(\text{Network Flow})$ which is $O(n^2 \log(n))$.

6 Mechanism Design in Practice

The designing of the Network Flow mechanism brings to attention the practicality of mechanism design in order to solve agent basepiped algorithmic problems. On one hand, we are
indeed able to create a mechanism that is able to solve this type of problem. On the other hand, the assumption in how much power we have over the design of the mechanism may not be representative of how much power we have over an actual system. For example, in order to ensure truthful disclosure of agents, we design the mechanism to be classified as a VGC mechanism. In particular, we enforce the constraint that \( o(t) = \arg\max_o \sum_i v^i(t^i, o) \).

In order to make this possible, we had to disregard all valuations of agents not immediately adjacent to the destination node in the network flow graph. It is hard to believe that as the designer of a mechanism, we can freely choose the value of independent agents. In many, and maybe even most cases, the valuation of an agent is private information. So being able to designate the valuation of an agent is not possible. More importantly, if we cannot enforce the valuations of the agents, we cannot guarantee that our objective function will be the sum of agent valuations. At that point we can no longer use the corrolary that agents will be truthful which VGC mechanisms promise.

Other problems with mechanism design come from the calculation of the output function \( o(t) \) and payment functions \( p^i(t) \). Given that agent valuations may be private, there may be no way to obtain these values. Since the payment and output functions are functions which depend on the valuations of each agent, there would be no way to calculate the payments to guarantee a system of truth telling agents.

Lastly, the algorithmic complexity of mechanism design depends on the complexities of the problem needed to be solved outside of the agent domain. As shown in the time analyses, the complexity comes out to be \( O(n \times O(\text{NonAgentAlgorithm})) \). If the complexity of the non agent algorithm is NP, mechanism design inherits this complexity.

Despite the criticisms of mechanism design, it is still a very powerful tool for utilitarian domains in which we are able to know agent valuations. This is especially relevent in cases like auctions in which money is an easily measured valuation. Additionally, if the agent’s type is related to its valuation, as is the case in auctions, and to an extent the shortest path problem, truthful disclosure of the type is another way to obtain the valuation of an agent.

### 7 Bibliography