Distributed Algorithmic Mechanism Design by Joan Feigenbaum, Michael Shapira, and Scott Shenker

Kristoffer Kalavainen

Master of Logic (ILLC)

October 2018 Amsterdam
Plan for talk

• What is Distributed Algorithmic Mechanism Design, or DAMD?
• A distributed VCG-mechanism.
• What is Interdomain Routing?
• Three perspectives on Interdomain Routing.
• Border Gateway Protocol.
• Main result.
Centralized vs. Distributed

Centralized Algorithm

Distributed Algorithm
Necessary Conditions for **AMD**

There are three necessary conditions for Algorithmic Mechanism Design:

1. There exists a trusted center, \( C \),
2. The information required to transmit to \( C \) is feasible,
3. The computational burden on \( C \) is manageable.
Let $T$ be a network, the **Network Complexity** includes (but is not limited to) these five quantities:

1. The computational burden on each node.
2. The maximal number of messages sent over any one edge.
3. The maximum size of a message.
4. The total number of messages sent over $T$.
5. The storage required at each node.
If there is no center performing the computation a network is susceptible to **Computational Manipulation**.

This includes:

i. Agents not communicating their computations.

ii. Agents communicating their computations incorrectly.

Let’s see an example!
A Distributed VCG mechanism

Let

- $O$ be a set of outcomes,
- $N$ a set of agents with valuation functions $v_i$,
- $\hat{o}$ be the outcome that maximizes social welfare and $W$ the value of this outcome:

\[
\hat{o} := \max \text{arg} \sum_{o \in O} \sum_{i \in N} v_i(o)
\]

\[
W := \max \sum_{o \in O} \sum_{i \in N} v_i(o)
\]

- Payment of agent $i$ be a function of $W$, $W_{-i}$ and $v_i$. 

Definition: An \textit{ex-post Nash equilibrium} is a strategy profile $s_1, ..., s_n$ such that for every $i$, all $\theta_1, ..., \theta_n$ and every $x'_i$ we have that

$$u_i(\theta_i, s_i(\theta_i), s_{-i}(\theta_{-i})) \geq u_i(\theta_i, x'_i, s_{-i}(\theta_{-i})).$$

Proposition Let $s_1, ..., s_n$ be an ex-post Nash equilibrium of a game $(A_1, ..., A_n; \Theta_1, ..., \Theta_n; u_1, ..., u_n)$. Define $A'_i = \{s_i(\theta_i) \mid \theta_i \in \Theta_i\}$, then $s_1, ..., s_n$ is a (weakly) dominant strategy equilibrium in the game $(A'_1, ..., A'_n; \Theta_1, ..., \Theta_n; u_1, ..., u_n)$. 
The problem of Interdomain routing is concerned with establishing connectivity over the Internet.

The Internet consist of many Autonomous Systems (AS), these

- Are frequently independent economic entities
- Want to minimize spread of private information
- May appear or disappear with short notice

⇒ **AMD does not work**
From the perspective of an AS a routing protocol should be able to handle

- Full connectivity between ASes
- Private routing preferences of ASes
- Quick reconfiguration in case an AS goes down or changes its preferences.
From a networking perspective there is a set of desiderata we want a routing protocol to satisfy:

- Distributive; because of (lack of) trust, scale and robustness.
- Adaptive; should work without *a priori* knowledge about the graphs’ topology.
- Efficient; it should not incur too high network complexity.
- Loop-free; even under autonomous routing preferences by nodes.
A **AS graph** is a tuple $G = (N, E)$, where $N$ is a set of nodes (corresponding to ASes) and $E$ a set of edges (or links) between these nodes.

Each node $i \in N$ has a set of forbidden paths. If a path is not forbidden, we say that it is permitted.

**Definition:** $P(i, d)$ denotes the set of *permitted, loop-free*, paths from $i$ to $d$. 
Definition: A routing protocol $T$ is a set of routes which, for each pair of nodes $i, d \in N$, contains at most one route $R_{i,d}^* \in P(i, d)$.

Definition: The route allocation $T_d$ is a subset of a routing protocol $T$ containing every route which ends in $d$.

Confluence of $T_d$

The route allocation

$$T_d = \{ R_1, R_2, ..., R_{n-1}, R_n \}$$

forms a confluent tree ending in $d$. 
Border Gateway Protocol (BGP) is the current standard for Interdomain routing on the Internet.

In BGP routes are stored as path-vectors, e.g. \((u_0, ..., u_k)\) denotes a path \(R_{u_0,d}\). Adding a new node to an AS Graph to the routing tree is done independently.

1. A node \(d \notin G\) wants to join the network.
2. \(d\) sends an update message to nearby nodes.
3. This update message propagates recursively throughout the graph.
More detailed, the routing process at a particular node $i$ follows this outline:

**Step 1.** An update message containing a route, $R_{i,d}$, to $d$ is received. If $R_{i,d}$ does not violate the policy of $i$ it is sent to the next step.

**Step 2.** If there are more than one route to $d$, then the node $i$ must eliminate all but one using its local preferences.

**Step 3.** If there is a change in preferred route to $d$, then $i$ sends an update message to its neighbours. First $i$ may filter, according to some set of export rules, which of its neighbouring nodes it will send an update message to.

Example!
Observation

BGP does not always converge on a route.

Definition: We say that a routing protocol, $T$, on a graph $G$ is robust if:

- $T$ converges, for any pair of nodes and,
- $T$ is convergent on every subgraph $H \subseteq G$. 
Let $R, S$ be two permitted routes on a graph $G$.

**Definition:** We say that $R \subseteq S$ if $R$ is a subpath of $S$ where both $S$ and $R$ end in the same node.

**Definition:** We say that $R \succ S$ if $\exists i \in N$ such that $R \succ_i S$ meaning that $i$ prefers $R$ over $S$.

Let $\emptyset$ be the transitive closure of both these relations, $\emptyset := (\subseteq \cup \succ)^*$.
Dispute wheel

Let $R$, $S$ be two permitted routes on a graph $G$.

**Definition:** We say that $R \subseteq S$ if $R$ is a subpath of $S$ where both $S$ and $R$ end in the same node.

**Definition:** We say that $R \succ S$ if $\exists i \in N$ such that $R \succ_i S$ meaning that $i$ prefers $R$ over $S$.

Let $\varnothing$ be the transitive closure of both these relations,

$$\varnothing := (\subseteq \cup \succ)^*.$$ 

**Definition:** If, for any two paths, $R \varnothing S$ and $S \varnothing R$ imply that $S$, $R$ start in the same node we say that $G$ has **no dispute wheel**.
Fact
If $G$ has no dispute wheel, then BGP is robust.
From a Mechanism-Design perspective consider the routing preferences of an AS as a selfish preference of an agent.

We want a routing protocol to satisfy:

- Efficiency; if implemented honestly it should be socially optimal.
- Incentive-compatibility; No agent should benefit by deviating from the protocol.

Remark
We will use maximization of social welfare as the socially optimal outcome.
We assign each node $i \in N$ a valuation function over each permitted route:

$$v_i : P(i, \cdot) \rightarrow \mathbb{R}_{\geq 0}$$

We require that $v_i(\emptyset) = 0$ and $v_i(R_{i,d}) \neq v_i(R'_{i,d})$ whenever $R \neq R'$. A route allocation $T_d$ maximizes social welfare if

$$T_d = \max_{T = \{R_{1,d}, \ldots, R_{d-1,d}\}} \sum_{i=1}^{N} v_i(R_{i,d})$$

Remark

This valuation function will correspond to the private type, $\theta$, of an agent.
Consider a computational network $G = (N, E)$ with a set of possible outcomes $O$. Each node $i$ has a private type $\theta_i \in \Theta$ and a utility function $u_i : \Theta \times O \rightarrow \mathbb{R}$.

**Definition:** A **distributed mechanism** $d^M$ is a tuple $d^M = (\Sigma, g, s^M)$ where:

- $\Sigma = (\Sigma_1, \ldots, \Sigma_n)$ is the set of feasible strategies
- $g : \Sigma \rightarrow O$ is the outcome function
- $s^M_i : \Sigma_i \times \Theta \rightarrow O_i$ is the prescribed strategy for agent $i$
**Definition:** A strategy profile $s^* \in \Sigma$ is an *ex-post Nash Equilibrium* of a distributed mechanism $d^M$ if,

for every agent $i$, every possible strategy $s'_i \in \Sigma_i$ and every $\Theta = (\theta_1, \ldots, \theta_n)$:

$$u_i(\theta_i, g(s^*_1(\theta_1), \ldots, s^*_n(\theta))) \geq u_i(\theta_i, g(s^*_1(\theta_1), \ldots, s'_i(\theta_i), \ldots, s_n(\theta_n)))$$

**Definition:** An ex-post Nash equilibrium is *collusion-proof* if every group deviation from the equilibrium ensures that at least one member is strictly worse off.
**Definition:** The set of valuation functions $\nu = \{\nu_1, \ldots, \nu_n\}$ on a computational graph is *policy consistent* if, for every two adjacent nodes $i, j \in N$ the extension of any two paths is monotone. That is to say, if $R, S \in P(j, d)$ and

$$v_j(R) \geq v_j(S) \text{ then } v_i((i, j) \cdot R) \geq v_i((i, j) \cdot S)$$
Main Theorem

Let $G$ be a graph where the valuation functions are policy consistent and do not induce a dispute wheel.

**Corollary:** In $G$, BGP converges to the socially optimal route allocation and is incentive compatible in collusion-proof ex-post Nash equilibrium (without monetary transfer).
Main Theorem

Let $G$ be a graph where the valuation functions are policy consistent and do not induce a dispute wheel.

**Corollary:** In $G$, BGP converges to the socially optimal route allocation and is incentive compatible in collusion-proof ex-post Nash equilibrium (without monetary transfer).

**Theorem:** In $G$, BGP converges to a unique stable, locally optimal route allocation $T^*_d$ for every $d \in G$

**Proof!**
Summing up

We saw:

- **DAMD** as a generalization of **AMD**.
- Network Complexity $\supset$ Computational Complexity.
- Computational Manipulation in an instance of **DAMD**.
- A sketch of Interdomain Routing with BGP.
- No Dispute wheels $\implies$ BGP is robust.
- Policy Consistency + No Dispute wheels $\implies$ locally optimal unique route allocation.
- Locally optimal unique route allocation $\implies$ collusion-proof ex-post Nash Equilibrium.
There is a known set of constraints on an AS graph which guarantee the absence of dispute wheels called the Gao-Rexford constraint.

We consider two types of relations on an AS graph, *Customer-Provider* and *Peer*.

For a given node $i \in N$, routes starting and terminating in nodes other from $i$ are called **transit routes**.

A graph $G$ adheres to the **Gao-Raxford Constraints** if

1. There are no Customer-Provider cycles.
2. Customers are strictly preferred over Peers.
3. Peers are strictly preferred over Providers.
4. Only allow transit routes for Customers.