T-79.194: Seminar on Theoretical Computer Science
A Study of Algorithms for Selfish Agents

Emil Falck <Emil.Falck@hut.fi>
49440c
Tik III

1 Introduction

The Internet is quickly becoming center of the global telecommunications infrastructure and the most important platform of computation. The resource sharing in the Internet occurs on several different levels and Internet users share access to the underlying transmission facilities themselves. With the best-effort nature of the Internet, where computers belong to different persons and organizations, resources are not reserved and all packets are serviced on a first-come-first-serve basis, the participating computers can no longer be expected to act as instructed. Aggressive and selfish computers and users get more than an equal share of the shared facilities. For example, web browsers that open more TCP connections receive more bandwidth as well as users that manipulate their TCP implementation to be less responsive when congestion is detected. One way to tackle the problem is to design the algorithms and protocols intended for Internet or any network with selfish computers to prevent this kind of behaviour.

In the paper a formal model is presented that allows studying algorithms that assume that the participants all act according to their own self-interest. The design problem is stated in the light of noncooperative game theory, from the view of a systems' engineer, who wishes to guide the global behaviour of the Internet. The selfishness is a problem that must be overcome with payments, trade and bribery.

The contibution of the paper is as follows: First a formal model for studying optimization problems is presented. The model is based on the field of mechanism design and was presented in [1]. A problem in this model has,in addition to the output specification, a description of the agents utilities. The mechanism has, in addition to the algorithm producing the desired output, payments to the participating agents.

Some solutions are provided with the known techniques from mechanism design for several basic optimization problems, in particular for the shortest path problem, where each edge may belong to a different agent.

Task scheduling and some more advanced prolems, which require new theory, are introduced and solutions to them are suggested.

2 The Model

In a way, mechanism design is the most real-world oriented topic in game theory. The problem involves designing a game where selfish agents will ultimately play according to the desire of the game designer. In this section the formal model is introduced for this purpose. The model was first represented in [2]. It is concerned with computing functions that depend on inputs that are distributed among n different agents. A problem in the model has both a specificiation of the function to be computed and a specification the goals of each agent. Thus the solution, called a mechanism, includes an algorithm computing the function as well as a strategy to hand out the payments to the agents. The idea of the payments is to motivate the agents to behave as intended. First a description of the mechanism design problem is presented, then a sound solution to the problem, an implementation with dominant strategies, is defined, and finally a special class of good solutions, truthful implementations, is introduced.

2.1 Mechanism design problem description

A mechanism design problem has two components: the algorithmic output specification, and the utility functions over the set of possible outputs describeing what the agents want.

Definition 1 (Mechanism Design Problem) ([1]) A mechanism design problem is given by an output specification and by a set of agent's utilities. Specifically:

1. There are n agents, each agent i has available to it some private input $t^i \in T^i$ (termed its type). Everything else in this scenario is public knowledge.
2. The output specification maps to each type vector $t = t^1 ... t^2$ a set of allowed outcomes o.
3. Each agent i's preferences are given by a real valued function: $v^i(o, t^i)$, called its valuation. This is a quantification of its value from the outcome o, when its type is t^i, in terms of some common currency. I.e. if the mechanism's outcome is o and in addition the mechanism hands this agent p^i units of this currency, then its utility will be $u^i = p^i + v^{7i}(o, t^i)$. This utility is what the agent aims to optimize.

The paper is limited to dicuss optimization problems. In these problems the outcome specification is to optimize a given objective function. The minimization problem is presented:

Definition 2 (Mechanism Design Optimization problem) ([1]) Mechanism Design Optimization problem is a mechanism design problem where the outcome specification is given by a positive real valued objective function $g(o, t)$ and a set of feasible outcomes F. The required output is the outcome $o \in F$ that minimizes g.

2.2 The Mechanism

Intuitively, given a problem, a mechanism solves it by making sure that the required outcome occurs, when agents choose their strategies as to maximize their own utilities.

Notation: $(a^1, ..., a^{i-1}, a^{i+1}, ..., a^n)$ is denoted by $a(^{-i})$. (a^i, a^{-i}) denotes the tuple $(a^1, ..., a^n)$.

Definition 3 (A Mechanism)([1]) A mechanism $m = (o, p)$ is composed of two elements: An outcome $o = o(a)$, and an n-tuple of payments $p^1(a)...p^n(a)$. Specificially:

1. The mechanism defines for each agent i a family of strategies A^i. Agent i can choose to perform any $a^i \in A^i$.
2. The first thing a mechanism must provide is an outcome function $o = o(a^1...a^n)$.
3. The second thing a mechanism provides is a payment $p^i = p^i(a^1...a^n)$ to each of the agents.
4. We say that a mechanism is an implementation with dominant strategies if
   - For each agent i and each t^i there exists a strategy $a^i \in A^i$, termed dominant, such that for all possible strategies of the other agents a^{-i}, a^i maximizes agent i's utility. I.e. for every $\hat{a}^i \in A^i$, if we define $o = o(a^i, a^{-i})$, $\hat{o} = o(\hat{a}^i, a^{-i})$, $p^i = p^i(a^i, a^{-i})$, $\hat{p}^i = p^i(\hat{a}^i, a^{-i})$, then $v^i(t^i, o) + p^i \geq v^i(t^i, \hat{o}) + \hat{p}^i$
   - For each tuple of dominant strategies $a = (a^i...a^n)$ the outcome o(a) satisfies the specification.

2.3 The Revelation Principle

The simplest types of mechanisms are those in which the agents' strategies are to simply report their types.

Definition 4 (Truthful Implementation) ([1]) We say that a mechanism is truthful if

1. For all i, and all t^i, $A^i = T^i$, i.e. the agents' strategies are to report their type. (This is called a direct revelation mechanism.)
2. Truth-telling is a dominant strategy, i.e. $a^i = t^i$ satisfies the definition of a dominant strategy above.

The revelation principle states that without loss of generality one can concentrate on truthful implementations.

Proposition 4.1 (Revelation Principle) ([3], p. 871) If there exists a mechanism that implements a given problem with dominant strategies then there exists a truthful implementation as well.

Proof: Formally, suppose $a^1(t^1), ..., a^n(t^n)$ implements mechanism $m = (o, p^1, ..., p^2)$ with dominant strategies. Consider $o^*(t^1...t^n) = o(a^1(t^1)...a^n(t^n))$ and $(p^*)^i(t^1...t^n) = p^i(a^1(t^1)...a^n(t^n))$.

The revelation prinicple says that the best strategy is for all agents to announce their utilities.

3 Applying Existing Mechanism Design Theory

In this section several well known mechanisms are presented. All the implementations are truthful, i.e. the following holds:

1. Each agent reports its input to the mechanism
2. The mechanism computes the desired outcome based on the reported types.
3. The mechanism computes payments for each agent.

The challenge is to determing the payments so that they ensure the truthful implementation.

3.1 Maximum

Story:

A single server is serving many clients. The server can serve exactly one request at a time. Each client has a private valuation t^i for his request being served. (The valuation is 0 if the request is not served.) We want the most valuable request to be served.

Failed attempts:

One might first attempt to simply ignore all payments setting $p^i = 0$ for all i. This however is clearly insufficient since it motivates each agent to exaggerate his valuation, as to get his request executed. The second attempt could be to let the winning agent pay his declaration, setting $p^i = -\hat{t}^i$ for the agent i with highest $\hat{t}^i andp^i = 0$ for the rest.. This is also insufficient since the agent with highest t^i is motivated to reduce his declaration to slightly above the second highest valuation. If agents have imperfect or incomplete information about others this strategic behaviour may lead to sub-optimal allocation.

Solution:

The agent that offers the highest valuation for his request pays the second highest price offered. I.e. $p^i = -t^j$, where i offers the highest price and j the second highest. All other agents have $p^k = 0$.

Analysis:

This is clearly a truthful implementation. Consider agent i and a lie $\hat{t}^i \neq t^i$. If thie lie does not change the allocation, then no harm is done and agent i's payment is unaffected. If thie lie gets his request served, then $\hat{t}^i > t^j > t^i$ and the agent gains t^i of utility from his valuation of the served request, but loses t^j on payments. Thus, his total utility would be $t^i - t^j < 0$, as opposed to 0 in the case of truth. On the other hand, if his lie makes him lose the service, then his utility is now 0, as opposed to a positive number which it was in the truthul case.

Comment: This is a typical auction and the solution presented if Vickrey's well-known second price auction. [5]

3.2 Construction of a public good

Story:

Consider a society of n citizens, and a public good G. The valuation of G is t^i for $i = 1, ..., n$. The cost of building the public good G is a publicly known constant C. The public good should be built only if $\Sigma_i t^i > C$. We want design a mechanism that charges the individuals in a way that we shall be able to determine if the sum of individual utilities is indeed greater than the cost of the public good, C. Note that we do not require to recover the cost of the cost of G.

Failed attempts:

One might first try to divide the total cost between the n citizens, i.e. set $p^i = -C/n$ for all i. Is this truthfully implementable? Assume that we have a bridge to build with $C = 8$. Suppose for example that $n = 2$. Cosider a society with two citizens 1 and 2 with $t^1 = 3$ and $t^2 = 6$. Thus, if the bridge is built, each citizen will pay $p_i = 4$. Then the bridge should be built because $\Sigma_i t^i = 9 > c = 8$. However, citizen 1 prefers to lie since $t^1 - p^1 = -1 < 0$. The citizen does not internalize the effects of his decision on the utilities of the other players. Likewise, this strategy prompts any citizen with $t^i > C/n$ to lie and announce his valuation as greater than C, thereby ensuring that the public good is built. Similary, if charge citizens the amount they declare (or something proportional to it), citizens could lie and report smaller valuations so as to reduce their payments. This may result in the wrong decision of not building G.

Solution:

We assert that the following mechanism solves the problem. Each citizen pays an amount $\max(0, C - \Sigma_{j \neq i} t^j)$, where t^j is the declared valuation of citizen j. That is, in case G is built, each citizen pays a sum equal to the minimum declaration required from him in order to build G, given the other's declarations.

Analysis:

This is a truthful implementation. To see this, suppose $\hat{t^j} > t^j$. Then the public good is built incorrectly if the lie $\hat{t^j}$ tips the sum of valuations above C, whereas the true worth t^j would have kept the sum below C. Then if the citizen would have played truthfully, his valuation would have been 0, but now it is $t^i - C + \Sigma_{j \neq \hat{t^j}} < 0$. Thus, truth would have been better. The case $\hat{t^j} < t^j$ is similar.

Comment:

The problem is well-known and the solution is known as the Clarke tax. [4]

Example:

3.3 Threshold

Strory:

A single cache is shared by many processors. When an item is entered into the cache, all processors gain faster access to this item. Each processor i will save t^i in communication costs if a certain item X is brought into the cache. (I.e. its valuation of loading X is $t^i > 0$, and of not loading it, 0.) The cost of loading X is a publicly known constant C. We want to load X iff $\Sigma_i t^i > C$.

Comment:

The problem is equal to the problem of Construction of a public good and is solved similarly.

3.4 Routing in a network - shortest path

Story:

Consider a network modeled by a directed graph, and two distinguished vetices x and y. The edges in the graph act as routers between the vertices they connect. Each router has a private cost for sending a message along it. The goal is to find the cheapest route x to y in the network. The set of feasible outcomes are all paths from x to y, and the objective function is the paths's total cost. Agent e's valuation is 0 if his edge is not part of the chosen path, and $-t^e$ if it is. For simplicity, the graph is assumed to be biconnected.

Solution:

Consider the following mechanism. The payment p^e given to agent e is 0 if e is not in the shortest path and $p^e = d_{G_{-e}} - (d_G - \hat{t}^e)$ if it is. Here $\hat{t}^e$ is the agents' reported input (which may be different from its actual one), d_G is the length of the shortest path (according to the inputs reported), and $d_{G_{-3}}$ is the length of the shortest path that does not contain e.

Analysis:

First notice if the same shortest path is chosen with $\hat{t}^e$ as with t_e, then the payment and thus utility of the agent does not change. A lie $\hat{t}^e > t^e$ will cause the algorithm to choose the shortest path that does not contain e as opposed to the correct one which does contain it iff $d_{G_{-e}} - d_g < \hat{t}^e - t^e$. This directly implies that e's utility would have been positive had e been chosen in the path (as opposed to 0 when its not), thus the truth is better. The case $\hat{t}^e < t^e$ is similar.

3.5 Utilitarian Functions

All previous examples are, in fact, Vickrey-Groves-Clark (VCG) mechanisms. In this section the general case is presented.

The VGC mechanism applies to mechanism design optimization problems where the objective function is simply the sum of all agents' valuations. The set of possible outputs is assumed to be finite.

Definition 5 ([1]) An optimization mechanism design problem is called utilitarian if its objective function satisfies $g(o,t) = \Sigma_i v^i(o,t^i)$.

Definition 6 ([1]) We say that a direct revelation mechanism $m = (o(t), p(t))$ belongs to the VCG family if

1. $o(t) \in arg\max_0(\Sigma_{i=1}^n v^i(t^i, o)$.
2. $p^i = \Sigma_{j\neq i} v^i(o(t), t^i) + h^i(t^{-i})$ where $h^i()$ is an arbitrary function of t^{-i}.

Theorem 5.1 A VCG mechanism is truthful.

Proof:

Let $d^1, ..., d^n$ denote the declaration of the agents and $t^1, ..., t^n$ denote their real types. Suppose that truth telling is not a dominant strategy, then there exists d,i,t,$\hat{d}^i$ s.t. $v^i(t^i, o(d^{-i}, t^i)) + p^i(t^i, o(d^{-i}, t^i)) + h^i(d^{-i}) < v^i(t^i, o(d^{-i}, \hat{d}^i)) + p^i(t^i, o(d^{-i}, \hat{d}^i)) + h^i(d^{-i})$ But then $\Sigma_{i=1}^n v^i(o(d^{-i}, t^i), t^i) < \Sigma_{i=1}^n v^i(o(d^{-i}, \hat{d}^i), t^i)$ in contradiction for the the definition of o().

Thus a VCG mechanism essentially provides a solution for any utilitarian problem. It is known that under mild assumptions VCG are the only truthful implementation for utilitarian problems.

4 Beyond Existing Mechanism Design

Many of the considerations of distributed computation are quite different from the ones usually considered in mechanism design, thus addressing them require new techinques and research. This section presents some scenarios in distributed computation that raise questions that go beyond the scope of traditional mechanism design.

4.1 Task Scheduling

Story:

A computer has k tasks it wishes to execute, and can execute each of them on any of n servers. Each server i knows, for every task j, the time t_j^i it requires to execute this task. Each server's cost is proportional to the time it spends on executing the tasks assigned to it. The goal is to have all tasks completed as soon as possible.

Issues:

Non-utilitarian Problem:

- **Non-utilitarian Problem:** The goal in this example is non-utilitarian. Thus, the VCG mechanism cannot be applied and new mechanisms need to be invented.
- **Impossibility:** It can be proved that no mechanism perfectly solves this problem. One should try to overcome this problem. The following approaches can be considered:
  1. **Approximation:** Find a mechanism that approximates the optimal solution. One could consider useing greedy algorithms, partial symmetry reductions, etc.
  2. **Randomization:** In Computer Science and in game theory randomization often helps. Monte Carlo algorithms are well studied and optimized. For instance, Metropolis Monte Carlo and Icing Model could be applied.
  3. **Model Extensions:** Every model is an imperfect abstraction of reality. Models can be extended or abstracted in order to get more tractable instances and better results.
- **Computational Intractbility:** Even from a purely algorithmic point of view, the task scheduling problem is NP-complete. When adding the requirements of a mechanism the problem gets evem harder. Standars ways pf overcoming the computational complexity have complicated interactions with the requirements of mechanism design.

4.2 Decentralized Auction

Story:

A single item is to be auctioned over the Internet among n humans, each with his own computer.

Restriction:

There is no trusted entity. The auctioneer is not trusted to faithfully follow the auction rules or to keep any secrets. In the absence of the trusted entity two goals should be ensured:

- The auction is executed according to the published auction rules (e.g. second price).
- No information about bids is leaked to the participants except for the identity of the winner, and the second highest bid.

Solution:

The "oblivious circuit evaluation" cryptographic protocols achieve the goal, as long as not too many participants lie. These protocols can faithfully carry out any distributed computation without leaking any information to the participants. On the other hand, the mechanism ensures that participants do not reveal tehir inputs. These protocols, while theoretically useful, are in practice too complex.

References

[1] N.Nisan. Algorithms for Selfish Agents. *(STACS 1999.)*

[2] N.Nisan, A.Ronen. Algorithmic mechanism design.

[3] A.Mas-Collel, W. Whinston, J. Green. Microeconomic Theory. Oxford university press, 1995.

[4] E.H.Clarke. Multipart pricing of public goods. Public Choice, pages 17-33, 1971.

[5] W.Vickrey. Counterspeculation, auctions and competitive sealed tenders. Journal of Finance, pages 8-37, 1961.

[6] David C.Parkes. Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency, Chapter 2: Classical Mechanism Design. (PdH Thesis, Dept of Computer and Information Science, University of Pennsylvania, May 2001).

[7] M.J.Osborne, A.Rubinstein. A course in game theory. MIT Press.