Incentive Analysis of Approximately Efficient Allocation Algorithms

(Extended Abstract)

Yevgeniy Vorobeychik* Yagil Engel†

ABSTRACT

We present a series of results providing evidence that the incentive problem with approximate *VCG*-based mechanisms is often not very severe. Our first result uses average-case analysis to show that if an algorithm can solve the allocation problem well for a large proportion of instances, incentives to lie essentially disappear. We next show that even if such incentives exist, a simple enhancement of the mechanism makes it unlikely that any player will find an improving deviation. Additionally, we offer a simulation-based technique to verify empirically the incentive properties of an arbitrary approximation algorithm and demonstrate it in a specific instance using combinatorial auction data.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics

General Terms

Economics

Keywords

Algorithmic mechanism design, combinatorial auctions, VCG

1. PRELIMINARIES

Let the utility function of a player be $u_i(t_i, o, p_i) = v_i(t_i, o) + p_i$, where $v_i(t_i, o)$ is the value that player i with type t_i has for outcome o, and p_i is his payment. A mechanism chooses an outcome o and assigns the payments p_i for all players i for a given joint report of types $t \in T$. Define *social welfare* to be $V(t, o) = \sum_{i \in I} v_i(t_i, o)$ and let $V^*(t) = \max_{o \in O} \sum_{i \in I} v_i(t_i, o)$ be the maximum welfare achieved for a type profile t. Let $V^* = \sup_{t \in T} V^*(t)$. It is well known that optimal allocation can be achieved as a truthful dominant strategy equilibrium by using Groves payments [4], with $p_i(t) = \sum_{j \neq i} v_j(t_j, o^*(t)) + h_i(t_{-i})$. Here h_i is an arbitrary function of the types reported by other players; for simplicity, we set it to 0.

Let $g : T \rightarrow O$ be an algorithm for computing approximately efficient allocation. Since g may compute only a suboptimal allocation, we let $V_g(t)$ be the welfare at the allocation $g(t)$, that is $V_g(t) = \sum_{i \in I} v_i(t_i, g(t))$. Define *VCG*-based payments by $p_i^g(t) = \sum_{j \neq i} v_j(t_j, g(t)) + h_i(t_{-i})$. Approximation algorithms typically include a guarantee with respect to the quality of approximation they provide. We say that $g(\cdot)$ is an α-approximation if $V^*(t) \leq \alpha V_g(t)$ for any type profile t.

2. CONNECTING APPROXIMATION AND INCENTIVES

Suppose that, somehow, we have an approximation bound for g that is a *known function* of $\alpha(t)$ for all $t \in T$. In the most trivial case, it could be just a fixed α, reducing the setup to the worst-case above. Alternatively, we may be able to split the set of type profiles into subsets $T^1, T^2, \ldots$, and obtain much better uniform bounds on some of these subsets than the worst case analysis would allow; for example, perhaps we know that for some large subset of combinatorial auction problems we can compute optimal allocation fast *exactly*, or nearly so. In any case, presently we will see that we need not even construct $\alpha(t)$ for all possible type profiles, but can obtain probabilistic bounds based on a subset of these.

THEOREM 1. *Suppose that the algorithm g is an $\alpha(t)$-approximation. Then truthful reporting constitutes an ϵ-Bayes-Nash equilibrium for $\epsilon = nE_t\left[\frac{\alpha(t)-1}{\alpha(t)} V^*(t)\right]$.*

3. APPLYING THE NON-UNIFORM INCENTIVE BOUND

A key question that stems from the above analysis is how a mechanism designer would determine an incentive bound for his algorithm in practice. We would not, for example, want to require the designer to obtain a non-trivial $\alpha(t)$ for every $t \in T$. Rather, we offer the following empirical approach.

1. Obtain or construct a simulator that allows one to sample joint player types $t \in T$ according to F
2. Collect a set of K joint type samples $t^1, \ldots, t^K$
3. For each t^k, compute $V_g(t^k)$ and $V^*(t^k)$ (or an upper bound on $V^*(t^k)$)

*University of Pennsylvania, Computer and Information Science Department; yev@seas.upenn.edu

†Technion, Industrial Engineering and Management; yagile@ie.technion.ac.il

Cite as: Incentive Analysis of Approximately Efficient Allocation Algorithms (Extended Abstract), Yevgeniy Vorobeychik and Yagil Engel, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1479-1480

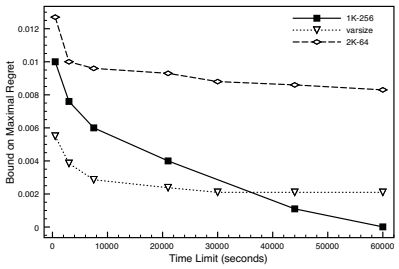

Figure 1: Worst-case bound of the regret, as a fraction of nV^*

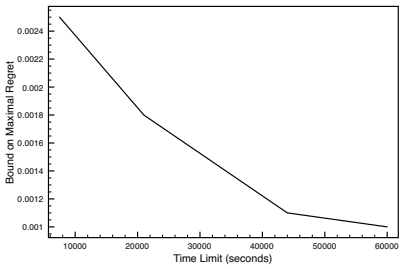

Figure 2: Worst-case bound of the regret for the union set of data, as a fraction of nV^*.

4. Compute $\alpha(t^k) = \frac{V^*(t^k)}{V_g(t^k)}$, let $\hat{Z}(t^k) = \frac{\alpha(t^k)-1}{\alpha(t^k)} V^*(t^k)$, and define $\hat{Z} = \frac{1}{K}\sum_{k=1}^{K} \hat{Z}(t^k)$
5. Compute a probabilistic bound based on $\hat{Z}$

For the last step, we assume that $\hat{Z}$ is Normally distributed (an assumption that is justified by the Central Limit Theorem when K is large), using $s^2(\hat{Z}(t^k))/K$ (where $s^2(\cdot)$ is the sample variance) as an estimate of the variance of $\hat{Z}$. Then, $E_t\left[\frac{\alpha(t)-1}{\alpha(t)}V^*(t)\right] \leq \hat{Z} + z_\delta\sqrt{\frac{s^2(\hat{Z}(t^k))}{K}}$ w.p. $\geq 1-\delta$, where z_δ is the value of Normal distribution at $1-\delta$.

3.1 Example: Combinatorial Auctions

To illustrate a concrete example applying the techniques introduced above, we now offer an incentive analysis of combinatorial auctions based on auction instances (in our notation, t^k) generated by CATS [3]. Since the absolute values of bounds are not very meaningful, and, further, CATS generates a set of bids, but does not specify the number of players (which could therefore be arbitrary), we opt to report bounds as multiples of nV^*. In doing so, we lose some tightness in the bounds, which we partially recover by letting $\hat{Z}' = \hat{Z}/\max_k V^*(t^k)$. Below we report $\hat{Z}'$.

The data set we used is composed of (a) a set of samples with 1000 bids on 144 goods ($1K-144$), (b) a set with 1000 bids on 256 goods ($1K-256$), (c) a set with 2000 bids on 64 goods ($2K-64$), and (d) a set with varying problem sizes (varsize). Each set contains 5000 samples, 500 for each of 10 different distributions. The data includes the result obtained by CPLEX which ran to optimality, the results obtained by CASS after about 7500 seconds for $1K-144$ and $1K-256$, or 44000 seconds for the other datasets, and, for the dataset $1K-256$, also the result obtained by the Gonen-Lehmann (GL) algorithm [2].

We computed the bound on regret for each dataset, as well as for the union set (named *all*). For each one we include the data for all CATS distributions except the arbitrary one. For $g(t)$ we used the following combination: we used the result returned by CPLEX for a sampled profile t^k if it was obtained in at most S seconds; otherwise the result returned by CASS was used. We varied the timelimit S is between 500 and 60000 seconds (about 16.6 hours). Such time limits could be reasonable for high volume auctions in which a lot of money is at stake.

In Figure 1, we show the resulting bound for each dataset as a function of the timelimit. It also quantifies the tradeoff between the amount of time given to the algorithm and regret (incentives for players to lie). The bounds are computed as explained above, with confidence level $1-\delta = 0.95$. The results are particularly encouraging for the union set (Figure 2), and $1K-256$, for which the regret approaches zero when the timelimit increases.

4. COMPUTING A BETTER RESPONSE

In this section we suggest a very simple sampling technique which allows us to amplify complexity of the deviation problem *on average*, under some assumptions on the algorithmic capabilities of the mechanism participants. Our first result reflects an assumption that the designer can construct a belief over the algorithms which would be used by each player. Under this assumption, let designer use his belief distribution over the algorithms to obtain L (random) instances of deviations t'_i for each i. Let $T'_i \subset T_i$ be the set of L such draws, as well as the reported type t_i. Let $g'_i(t) = \arg\max_{o=g(t'_i,t_{-i})|t'_i\in T'_i} \sum_{j\in I} v_j(t_j, g(t'_i, t_{-i}))$ and let $g'(t) = \arg\max_{i\in I} g'_i(t)$.

THEOREM 2. *Given $g'(t)$ as the allocation mechanism, the probability that some player can compute an improving deviation is at most $\frac{n}{L+1}$.*

While the assumption that algorithmic capabilities of players are predictable is often reasonable, we may wish to make a stronger statement. Let $G(u)$ be the distribution function of player utilities induced by the *designer*'s search process (e.g., uniform sampling from the type space), whereas $H(u)$ is the distribution function of player i's utilities induced by the player's search.

THEOREM 3. *Let $U_1 = \{u|G(u) = 1\}$ and suppose that $H(U_1) = 0$. Then $\lim_{L\to\infty} \int G(u)^L dH(u) = 0$.*

The interpretation is that as long as the players do not have a positive probability of reaching a utility that is better than *any* that the designer *can possibly* attain, the designer can use random sampling to effectively eliminate incentives to lie.

5. REFERENCES

[1] Peter Cramton, Yoav Shoham, and Richard Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.
[2] R. Gonen and D. Lehmann. Linear programming helps solving large multi-unit combinatorial auctions. In *Electronic Market Design Workshop*, 2001.
[3] Kevin Leyton-Brown and Yoav Shoham. A test suite for combinatorial auction. In Cramton et al. [1], chapter 18, pages 451–478.
[4] Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford Univ. Press, 1995.