Tractable Computational Methods for Finding Nash Equilibria of Perfect-Information Position Auctions

David Robert Martin Thompson
Department of Computer Science
University of British Columbia
201-2366 Main Mall, BC V6T 1Z4, CANADA
daveth@cs.ubc.ca

Kevin Leyton-Brown
Department of Computer Science
University of British Columbia
201-2366 Main Mall, BC V6T 1Z4, CANADA
kevinlb@cs.ubc.ca

ABSTRACT

Due to the economic importance of the position auctions used by search engines to sell advertising, these auctions have received considerable recent study. However, most of this study has been analytic, and these analyses have relied on strong assumptions about the structure of the setting. In this paper, we show that it is feasible to perform *computational* equilibrium analyses of complex, realistic auction problems like advertising auctions. In particular, we show for the first time that the Nash equilibria of position auctions can be computed exactly, and we do so without relying on any of the assumptions that are necessary for closed-form analysis. We achieve these results by deriving a polynomial-sized action graph game representation of the position auction (discretizing bid amounts) and then finding a Nash equilibrium of that game. Our formulation makes it possible to show how equilibrium behavior, revenue and efficiency vary across auction types (Generalized First Price or Generalized Second Price), payment structures (pay-per-click or pay-per-impression), and click-through bias (position does/does not also depend on advertiser's click-through rate).

1. INTRODUCTION

Modern search engines derive their revenue from contextual advertising: given a user's keyword search, the engine provides relevant ads alongside the results. The search engine chooses which ads to show based on the results of "position auctions" that are conducted automatically every time a search is performed. In a position auction, higher bids are awarded higher positions in the list of ads, but also made to pay larger amounts. In what follows we explain position auctions, describing different versions of auction type (GFP/GSP), payment structure (pay-per-click/pay-per-impression) and click-through bias (position does/does not also depend on advertiser's click-through rate)

First, we describe auction type. The two most common types of position auctions are *generalized first-price* and *generalized second-price*; these are the types upon which we focus in this paper. Both auction types ask bidders to submit single-value bids and then rank the bids from highest to lowest, awarding the highest bidder the highest position, the second-highest bidder the second-highest position, and so on. The auction types differ in the way that bidders are charged. In a generalized first-price auction (GFP) each bidder who wins a position pays the amount that he bid. In a generalized second-price auction (GSP), on the other hand, each bidder pays the minimum amount that he would have had to bid to maintain his awarded position. (For example, the highest bidder pays the amount of the second-highest bid, because if he had bid less than this amount, he would have received the second position instead of the first.) The earliest position auctions deployed commercially were GFP, but all of the major search engines have since adopted GSP.

Second, position auctions differ according to whether they require bidders to pay per impression or pay per click. In a pay-per-impression auction, the bidder must pay every time his ad is shown, while in a pay-per-click auction, the bidder only pays when a user clicks on his ad. Search engines use pay-per-click auctions, though pay-per-impression advertising is used in other online contexts such as banner ads.

Finally, some ads are more frequently clicked on than others. Some search engines bias their auction rules to take this fact into account. Specifically, they assign each ad a "quality score" that reflects its likelihood of receiving a click, and rank bids by the product of the bid and the quality score (i.e., by the expected revenue that the ad will yield) rather than by the bid amount alone. We refer to auctions that rank by quality score as *weighted* auctions, to auctions that rank only by bid amount as *unweighted auctions*, and to the product of a bid and a quality score as an *effective bid*.

1.1 AGGs

Since analyses of position auctions have relied on perfect-information games, in principle we could discretize bid amounts and then represent the auction as a normal-form game. This would have the advantage that Nash equilibria of the auction could be identified by standard computational tools such as Gambit [22]. The catch, of course, is that the normal form representation of a realistic ad auction problem is unmanageably large. For example, the normal form representation of a relatively small game with 10 agents and 10 bid amounts per agent consists of 100 billion values, too many to store even on the hard drive of many modern computers—let alone in RAM. Thus, to have any hope of tackling position auctions computationally, it is necessary to work with a representation language that allows the game to be compactly described.

We chose to use the action-graph game (AGG) representation [5, 15]. Action-graph games are similar to the more widely-known graphical game representation [16] in that they exploit utility independencies. AGGs are strictly more powerful than graphical games, however. This is because AGGs are compact not only for games with "strict utility independencies" (the property that one agent's payoff never depends on some second agent's action) but also "context-specific independencies" (one agent's payoff is independent of a second agent's action, at least for some action of the first agent and some set of actions of the second). This distinction is important for modeling position auctions. Note that any bidder can affect any other bidder's payoff (e.g., by outbidding him); hence the graphical game representation of a perfect-information position auction is a clique, meaning that it is no more compact than the normal form. However, position auctions have considerable context-specific independence structure. To give one simple example, in a GFP auction, bidder i's utility is independent of bidder j's bid, conditional on j bidding less than i. This is the sort of structure that can be captured by AGGs.

The core idea behind action-graph games is the action graph, so-called because nodes in this directed graph represent actions. Each agent is allowed to choose his action from an arbitrary subset of the nodes; crucially, agents' subsets are allowed to overlap or coincide. Play of the game can be visualized as each agent simultaneously placing a single token on one of the nodes in the graph. Given the locations of all the tokens, an agent's utility can be computed by referring only to the *number* of tokens in the *neighborhood* of his chosen node. (The neighborhood of a node v is the set of all nodes having outgoing edges that point to v; self-edges are allowed, and so a node can belong to its own neighborhood.) Figure 1 gives an example of an action-graph game taken from [5]. Observe that there are two action sets consisting of four actions each; unlike in a graphical game, the number of agents cannot be inferred from the graph.

Their compact size is not the only interesting thing about AGGs. More importantly, AGG structure can be leveraged computationally, and hence game-theoretic computations can be performed dramatically more quickly for AGGs than for games represented in normal form. For example, given action graphs with bounded in-degree, a polynomial-time dynamic programming algorithm can be used to compute an agent's expected utility under an arbitrary mixed strategy profile [15]. (Observe that this is interesting because the standard method of computing expected utility has running time polynomial in the size of the normal form, but potentially exponential in the size of more compact representations like AGGs.) This computational problem is important because it constitutes the inner loop of many game-theoretic algorithms, including state-of-the-art algorithms for computing Nash equilibria like Simplicial Subdivision [26] and Govindan-Wilson [14]. This implies that an exponential speedup to the solution of the expected utility problem translates directly to an exponential speedup of such algorithms, without any effect on the solution obtained.

Although it is beyond the scope of this paper to describe AGGs in detail, there *is* one further element of the representation that we must describe here. Specifically, it is possible to add so-called *function nodes* to the action graph, which are nodes that belong to no agents' action sets. Instead, the "action count" at a function node is calculated as an (arbitrary) deterministic function of the counts at the function node's parents. For example, when an agent's payoff for playing a depends on how many agents play any of b, c or d, we can add a summation node to reduce the in-degree of a. Function nodes can dramatically reduce representation size when (for example) many actions affect a given action in the same way. As long as the functions are well-behaved ("contribution-independent"; roughly, commutative and associative) function nodes can be used with the dynamic programming algorithm from [15]. Since the variable most important to the asymptotic running time of this algorithm is the maximal in-degree of action nodes, and since this quantity can be drastically reduced by the introduction of high-in-degree function nodes, function nodes can also lead to substantial computational savings.

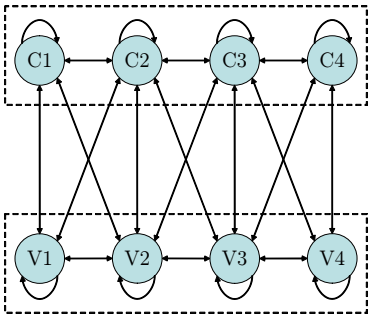

Figure 1: Taken from [5]. The ice-cream vendor game as an AGG. Chocolate and vanilla ice-cream vendors must choose which block to open their stores on, given that their payoffs will depend on how many other vendors are within one block of their location. (Circular nodes represent actions, dotted boxes represent action sets for a group of players and arcs represent payoff dependencies.)

2. MODEL AND REPRESENTATION

2.1 Ad auction setting

In our evaluation of auction types, we use the following model. As in previous work [28, 11, 18], we look for the equilibria of the one-shot, full-information game.

DEFINITION 1 (AD AUCTION SETTING). *An* ad auction setting *is given by a 6-tuple* (N, K, M, P, V, β)*:*

1. *N is the set of bidders;*
2. *K is the set of possible positions;*
3. *M is the set of possible bids;*
4. *P is a matrix of click-through rates, where $P_{j,i}$ is the probability that bidder i will receive a click when his ad is in position j;*
5. *V is a matrix of expected values for a click, where $V_{j,i}$ is the value that bidder i has for a click when his ad is in position j; and*

6. *β is a vector of "quality scores" or weights, where β_i/β_j denotes the quality of bidder i relative to bidder j.*

2.2 Representing GFPs as Action-Graph Games

Having described the auction types and model that we intend to solve, our next step is to show that they can be compactly represented. This section will present algorithms for representing an ad auction type (for example, weighted per-click GSP) and auction setting as an action-graph game and bounds on the size of those representations.

To get a suitably compact representation, one of our biggest concerns is the maximum in-degree of our action graph. Every action node must have a table representing that action's payoff function, and this table will grow exponentially in the in-degree of that action node. An arc (u, v) in an AGG denotes that the payoff for playing action v depends on the number of agents playing u. However, note that an agent's payoff depends only on the position he is awarded and the price he is made to pay. In the case of a GFP, an agent's price is determined by his bid while his position is determined by the number of bids above his and the number of bids equal to his. (We assume that ties are broken randomly.) His position can be affected by many different actions by other bidders, leading to a very large in-degree. However, if we introduce function nodes corresponding to summation (we will call these *summation nodes*) to keep track of how many bids are equal to or greater than each possible bid value, we need only two in-arcs to capture the two values.

Algorithm 1 converts a weighted (or, as a special case, unweighted[1]) GFP to an AGG. An example AGG is shown in Figure 2.

foreach agent $i \in N$ **do**
  foreach bid $m \in M$ **do**
    create an action node representing i bidding m;
$E \leftarrow \{m\beta_i | \forall i \in N, \forall m \in M\}$;
foreach effective bid $e \in E$ **do**
  create a summation function node, $(=, e)$ representing the bidders bidding exactly e;
  create a summation function node, $(\geq, e)$ representing the bidders bidding above e;
  add an arc from $(=, e)$ to $(\geq, e)$;
  if *e>0* **then**
    add an arc from $(\geq, e)$ to $(\geq, e')$ (where e' is the next largest effective bid);
foreach action node a **do**
  $e \leftarrow$ effective bid of a;
  add an arc from a to $(=, e)$;
  add an arc from $(=, e)$ to a;
  add an arc from $(\geq, e)$ to a;

Algorithm 1: An algorithm for converting an auction setting into an action graph representing a GFP.

For each action node, we must have a payoff function mapping from the inputs to that node to the payoff an agent playing that action will get. For a bid of b by agent i, we denote this as $\gamma_{i,b}^{1,I}$ for a pay-per-impression GFP. Because of the configuration of the summation nodes, the two inputs to this function are the number of effective bids that are equal to i's bid of b and the number that are greater than or equal. The payoff function for any action node in a pay-per-impression GFP is given by

$$\gamma_{i,b}^{1,I}(e, g) = \frac{1}{e} \sum_{j=g-e+1}^{\min(g,k)} (P_{j,i} V_{j,i} - b).$$

Similarly, the payoff function for any action node in a pay-per-click GFP is given by

$$\gamma_{i,b}^{1,C}(e, g) = \frac{1}{e} \sum_{j=g-e+1}^{\min(g,k)} P_{j,i}(V_{j,i} - b).$$

This representation results in a graph containing nm action nodes, each of which has an in-degree of two. Each node has a payoff table with at most $O(n^2)$ relevant entries. Thus, this representation requires $O(n^3 m)$ space.

2.3 Representing GSPs as Action-Graph Games

GSPs are similar to GFPs in that each agent's payoff depends on a small number of values. To determine the position (or range of positions), we use the exact same graph structure as for GFPs. However, we need to augment the graph to capture the pricing rule of GSPs. This is done by adding "price nodes", function nodes that identify the next-highest bid. We use the term *argmax node* to refer to a function node whose value is equal to the largest (given some arbitrary ordering) in-arc carrying a non-zero value. By ordering action nodes according to the value of their effective bids, an argmax node identifies the highest effective bid among the subset of action nodes connected to it. After running the Algorithm 1, we add argmax nodes as shown in Algorithm 2. An example of the resulting action graph is illustrated in Figure 3. Note that although the in-degree of the argmax nodes can get large ($O(nm)$), the computational complexity of solving an AGG only depends on the in-degree of the action nodes.

foreach effective bid $e \in E$ **do**
  create an argmax function node, (p, e) representing the next highest effective bid below e;
  foreach action node a with effective bid e' **do**
    if *e' < e* **then**
      add an arc from a to (p, e);
    if *e' = e* **then**
      add an arc from (p, e) to a;

Algorithm 2: An algorithm for converting an auction setting into an action graph representing a GSP.

As in the case of GSP, we must define a payoff function for each action node. Now, we have a third input (p), which identifies the next-highest effective bid. Let E_p denote this value. The payoff function for any action node in a pay-per-impression GSP is given below.[2]

[1]Trivially, this algorithm can also be used to represent an unweighted GFP by replacing β with a vector of ones, causing the auction to treat bids by different agents equivalently.

[2]$\delta(x) = 1$ iff x is true.

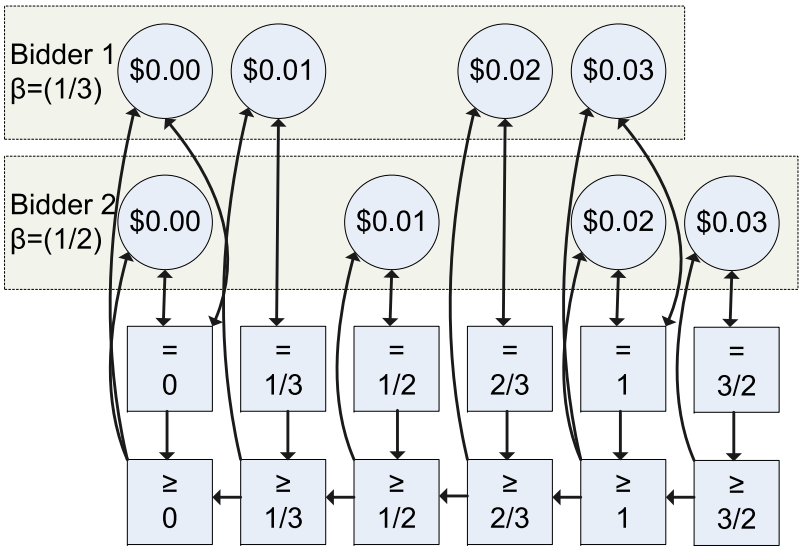

Figure 2: A weighted GFP represented as an AGG. (Square nodes represent summation function nodes.)

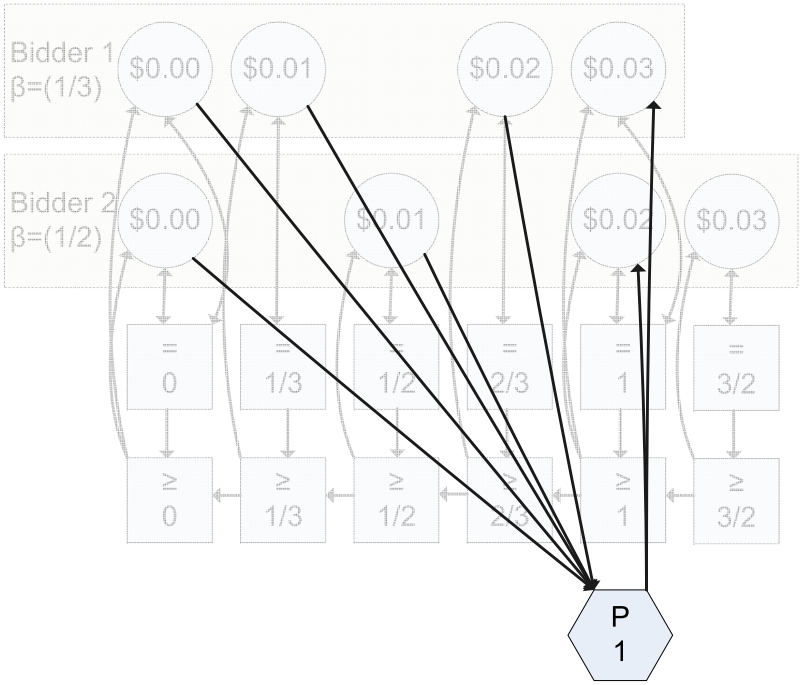

Figure 3: To represent a GSP as an AGG, we add price nodes (argmax nodes denoted by hexagons) to a GFP representation. For clarity only one price node is pictured, while a full GSP representation requires one price node for each effective bid.

$$\gamma^{2,I}_{i,b}(e,g,p) = \frac{1}{e}\sum_{j=g-e+1}^{\min(g-1,k)}(P_{j,i}V_{j,i}-b) + \delta(g \le k)(P_{j,g}V_{j,g}-E_p)/e$$

Similarly, the payoff function for any action node in a pay-per-click GFP is given by

$$\gamma^{2,C}_{i,b}(e,g,p) = \frac{1}{e}\sum_{j=g-e+1}^{\min(g-1,k)}P_{j,i}(V_{j,i}-b) + \delta(g \le k)P_{j,g}(V_{j,g}-E_p)/e$$

This representation results in a graph containing nm action nodes, each of which has an in-degree of three. Each node has a payoff table with at most $O(n^2|E|)$ relevant entries (where $|E| \le nm$). Thus, this representation requires $O(n^4m^2)$ space. As was the case with GFPs, we can produce an unweighted auction by simply replacing β with a vector of ones. We can find also AGG representations of Lahaie and Pennock's family of ranking rules [19] by adjusting the values of β appropriately.

In summary, we model eight auction types by the following methods:

1. unweighted, pay-per-impression GFP: replace β with a vector of ones, generate graph according to Algorithm 1, $\gamma^{1,I}$ payoff function
2. weighted, pay-per-impression GFP: generate graph according to Algorithm 1, $\gamma^{1,I}$ payoff function
3. unweighted, pay-per-click GFP: replace β with a vector of ones, generate graph according to Algorithm 1, $\gamma^{1,C}$ payoff function
4. weighted, pay-per-click GFP: generate graph according to Algorithm 1, $\gamma^{1,C}$ payoff function
5. unweighted, pay-per-impression GSP: replace β with a vector of ones, generate graph according to Algorithm 1 followed by Algorithm 2, $\gamma^{2,I}$ payoff function
6. weighted, pay-per-impression GSP: generate graph according to Algorithm 1 followed by Algorithm 2, $\gamma^{2,I}$ payoff function
7. unweighted, pay-per-click GSP: replace β with a vector of ones, generate graph according to Algorithm 1 followed by Algorithm 2, $\gamma^{2,C}$ payoff function
8. weighted, pay-per-click GSP: generate graph according to Algorithm 1 followed by Algorithm 2, $\gamma^{2,C}$ payoff function

3. EXPERIMENTAL SETUP

3.1 Computational Environment

Our experiments were performed using a computer cluster consisting of 55 machines with dual Intel Xeon 3.2GHz CPUs, 2MB cache and 2GB RAM, running Suse Linux 10.1. All timing results given below report CPU time (rather than wall clock time). To compute Nash equilibria, we used the simplicial subdivision [26] implementation provided by Gambit [22], extended to use AGGs and the dynamic programming algorithm of [15].[3]

3.2 Problem Distribution

To evaluate the empirical performance of our approach, we generate instances from a distribution over auction settings. The size of the problem is specified by n and m where $M = \{x|x \in \mathbb{Z}, 0 \le x \le m\}$, i.e. users can bid any integer up to m. For each bidder i, we generated a bidder-specific click-through factor $b_i \sim$ uniform$[0,1]$. For each position j (up to $k = \min(n/2, 8)$), we generate a position-specific click-through factor e_j where $e_1 \sim$ uniform$[0,1]$ and $e_j \sim$ uniform$[0, e_{j-1}]$. Each entry in click-through-probability matrix P, is a product of these factors ($P_{j.i} = \beta_i e_j$). For each bidder, we generate a single value per click (which is the same for all positions) drawn from uniform$[0,1]$. These values are renormalized so that the highest value is equal to m (these values are normalized per-click or per-impression depending on which type of auction we evaluate). This model is a encodes common assumptions such as separability [28],

3.3 Experimental Results: Computational

3.4 Representation Size

We compare the our representation size against the corresponding normal form games in Figure 4 and Figure 5. The size of the normal form game was computed using n 32-bit values per table entry. GSPs are much larger than GFPs because they have higher-dimensional payoff functions. Weighted auctions are smaller than unweighted because they have far fewer ties (meaning the payoff function needs to be evaluated in fewer places). Because pay-per-click and pay-per-impression have identical graph structures and identical potential for ties, they result in the same representation size.

3.5 Performance

To evaluate the empirical hardness of solving our AGG representations, we drew 200 samples from the problem distribution described above (for $n = 10, k = 5, m = 10$). For each instance, we represented all eight auction types as AGGs and ran the solver on them for 120 seconds each. The results of this experiment are shown in Figure 6. Somewhat surprisingly, GFPs are appreciably harder to solve than GSPs, despite their smaller input size. Emperically, simplicial subdivision often performs faster when solving games with small-support equilibria. Given that GFPs tend to give rise to oscillating equilibria in the repeated case [3, 10], the one-shot game may only have large-support equilibria.

[3]The extension of Gambit that allows simplicial subdivision to work with AGGs can be obtained from `http://cs.ubc.ca/~jiang/aggsoft`.

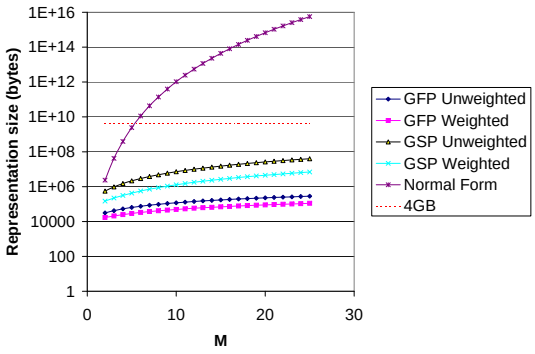

Figure 4: Representation size as a function of the number of bid increments. $(n = 10, k = 5)$

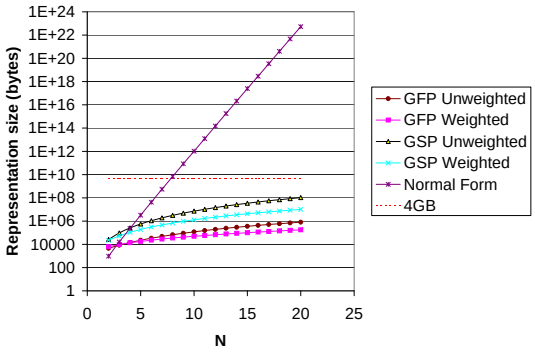

Figure 5: Representation size as a function of the number of agents. $(m = 10, k = \lfloor n/2 \rfloor)$

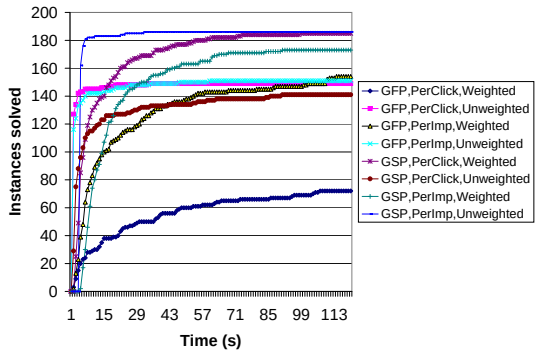

Figure 6: Fraction of problems solved in under t seconds for each of the auction types.

4. EXPERIMENTAL RESULTS: ECONOMIC

4.1 Weak Equilibria

To get meaningful results, we need to consider the problem of weak equilibria. Auctions frequently have many equilibria that can differ in revenue and economic efficiency. In the case of GSPs, each agent can vary his bid within a range that does not affect the ranking of the ads, but does affect the amount that the bidder above must pay. [6, 28] Thus, the equilibrium revenue of GSP depends on which strategies each agent chooses from a set of weak best responses. A similar problem arises when we consider the support size of an equilibrium. Our solver could converge to an equilibrium where agents mix needlessly across the range of best response bids.

To address this problem, we give the bidders a small bias, favoring some bids over others. This is done by perturbing each agent's utility function by a small (10e-6) amount in favor of low (or high) bids regardless of the outcome.

4.2 Comparing Auction Types

To explore the differences between different auction types, we drew 200 samples from the setting distribution. For each of these settings, we represented all eight auction types, with high, low and no bidding bias (24 games per setting) and computed their equilibria. The following results are based on the 2783 (of 4800) games for which we could compute an equilibrium in under one CPU hour.

For each Nash equilibrium, we computed the expected revenue (see Figure 7) normalized relative to the revenue of the truthful equilibrium of VCG. A number of trends are apparent from this plot. GSPs are more sensitive to equilibrium selection. Unweighted pay-per-impression auctions outperform weighted pay-per-impression auctions while weighted pay-per-click auctions outperform unweighted pay-per-click auctions. This might explain why (with the exception of Yahoo!'s unweighted pay-per-click GSP) these auctions are not found in practice. When bids are discrete, VCG revenue is not a lower bound on the revenue of weighted pay-per-click GSPs.

We can also compare the expected social welfare generated by an auction. (Note that this is the social welfare of the advertisers.) As was the case with revenue, unweighted pay-per-impression auctions outperform weighted pay-per-impression auctions while weighted pay-per-click auctions outperform unweighted pay-per-click auctions. Despite the coarse bidding language, weighted pay-per-click GSPs find the optimal allocation with extremely high probability. In the case of unweighted pay-per-click GSPs, different bidding biases lead to equilibria with significantly different social welfare. This indicates that the allocations of such auctions are sensitive to some bidder's choice among weak best responses.

Lastly, we can consider what kinds of equilibria these different auction types give rise to. As described above, GFPs seem to lead to a lot of mixing in equilibrium. To quantify this we measure the Shannon Entropy[27] of the joint strategy (see Figure 9). Given random variable X with distribution $p()$, the Shannon Entropy is $H(X) = -\sum p(x) \log_2(p(x))$. This gives a measure of the uncertainty associated with X, in this case the joint action of the bidders. Our biasing

approach seems to be successful: the equilibria we found in biased games have less entropy than the equilibria of the unbiased games. However, unweighted GFPs have much higher entropy, regardless of biasing, which might explain why their Nash equilibria were harder to find.

5. CONCLUSIONS AND FUTURE WORK

For the first time, our work shoes that position auctions, although difficult to analyze manually, are within reach of modern computational methods. We show how the results of these computational methods can be used to compare the economic properties of different auction types.

There is opportunity for future work both on the preferences and on the computational methods. Within our model of auction settings, we could immediately explore alternative preference distributions that relax common assumptions (such as separability of click-through-rates). Also, better distributions over preferences could be learned from real-world bidding data. There is also the question of how to compactly represent games given a richer preference model. (For example, we could model settings where an ad's click and conversion probabilities depends on which ads are shown above it.) Better computational methods could also improve this approach: by using an equilibrium-finding algorithm that allows us to control which part of strategy space is explored first (such as the support enumeration methods of [25]), we could find desired types of equilibria without perturbing the payoffs of the game.

6. ACKNOWLEDGMENTS

This work was supported by a grant from Microsoft.

7. REFERENCES

[1] Z. Abrams, O. Mendelevitch, and J. Tomlin. Optimal delivery of sponsored search advertisements subject to budget constraints. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, pages 272–278, 2007.

[2] G. Aggarwal, A. Goel, and R. Motwani. Truthful auctions for pricing search keywords. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, pages 1–7, New York, NY, USA, 2006. ACM.

[3] K. Asdemir. Bidding patterns in search engine auctions. In *Second Workshop on Sponsored Search Auctions*, 2006.

[4] I. Ashlagi, D. Monderer, and M. Tennenholtz. Mediators in position auctions. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, pages 279–287, New York, NY, USA, 2007. ACM.

[5] N. Bhat and K. Leyton-Brown. Computing Nash equilibria of Action-Graph Games. In *UAI: Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 35–42, 2004.

[6] T. Borgers, I. J. Cox, M. Pesendorfer, and V. Petricek. Equilibrium bids in auctions of sponsored links: Theory and evidence, as of November 2006.

[7] C. Borgs, J. Chayes, N. Immorlica, K. Jain, O. Etesami, and M. Mahdian. Dynamics of bid optimization in online advertisement auctions. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 531–540, New York, NY, USA, 2007. ACM.

[8] N. Buchbinder, K. Jain, and S. Naor. Online primal-dual algorithms for maximizing ad-auctions revenue. In *Proceedings of the 15th Annual European Symposium on Algorithms*, 2007.

[9] M. Cary, A. Das, B. Edelman, I. Giotis, K. Heimerl, A. R. Karlin, C. Mathieu, and M. Schwarz. Greedy bidding strategies for keyword auctions. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, pages 262–271, New York, NY, USA, 2007. ACM.

[10] B. Edelman and M. Ostrovsky. Strategic bidder behavior in sponsored search auctions. *Decis. Support Syst.*, 43(1):192–198, 2007.

[11] B. Edelman, M. Schwarz, and M. Ostrovsky. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, March 2007.

[12] J. Feng, H. K. Bhargava, and D. Pennock. Comparison of allocation rules for paid placement advertising in search engines. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, pages 294–299, New York, NY, USA, 2003. ACM.

[13] J. Feng, H. K. Bhargava, and D. M. Pennock. Implementing sponsored search in web search engines: Computational evaluation of alternative mechanisms. *INFORMS J. on Computing*, 19(1):137–148, 2007.

[14] S. Govindan and R. Wilson. Essential equilibria. *Proceedings of the National Academy of Sciences USA*, 102:15706–15711, 2005.

[15] A. X. Jiang and K. Leyton-Brown. A polynomial-time algorithm for Action-Graph Games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 679–684, 2006.

[16] M. Kearns, M. Littman, and S. Singh. Graphical models for game theory. In *UAI: Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2001.

[17] B. Kitts, P. Laxminarayan, B. LeBlanc, and R. Meech. A formal analysis of search auctions including predictions on click fraud and bidding tactics. In *Workshop on Sponsored Search Auctions*, 2005.

[18] S. Lahaie. An analysis of alternative slot auction designs for sponsored search. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, pages 218–227, New York, NY, USA, 2006. ACM.

[19] S. Lahaie and D. M. Pennock. Revenue analysis of a family of ranking rules for keyword auctions. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, 2007.

[20] S. Lahaie, D. M. Pennock, A. Saberi, and R. Vohra. Sponsored search auctions. In Nisan et al. [24], chapter 28, pages 699–719.

[21] M. Mahdian, H. Nazerzadeh, and A. Saberi. Allocating online advertisement space with unreliable estimates. In *EC: Proceedings of the ACM Conference on Electronic Commerce*, pages 288–294, New York, NY, USA, 2007. ACM.

[22] R. D. McKelvey, A. M. McLennan, and T. L. Turocy. Gambit: Software tools for game theory, 2006. `http://econweb.tamu.edu/gambit`.

[23] A. Mehta, A. Saberi, U. Vazirani, and V. Vazirani. Adwords and generalized on-line matching. In *FOCS '05: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, pages 264–273, Washington, DC, USA, 2005. IEEE Computer Society.

[24] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, Cambridge, UK, 2007.

[25] R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 664–669, 2004.

[26] H. Scarf. The approximation of fixed points of continuous mappings. *SIAM Journal of Applied Mathematics*, 15:1328–1343, 1967.

[27] C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[28] H. Varian. Position auctions. *International Journal of Industrial Organization*, 25(6):1163–1178, 2007.

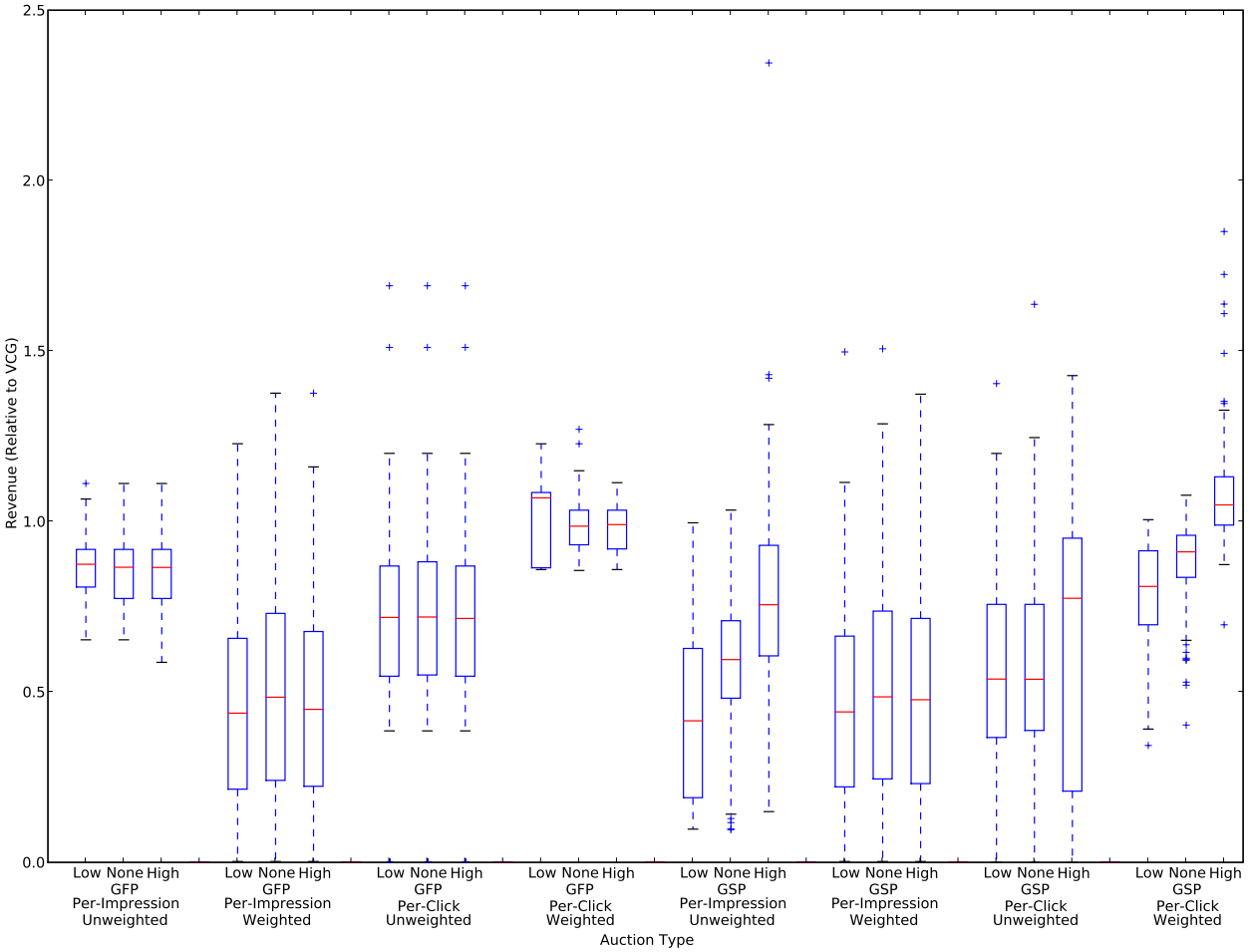

Figure 7: Expected revenue in equilibrium under different auction types. The three columns for each auction type represent the different equilibrium biases: low bidding, no bias and high bidding from left to right.

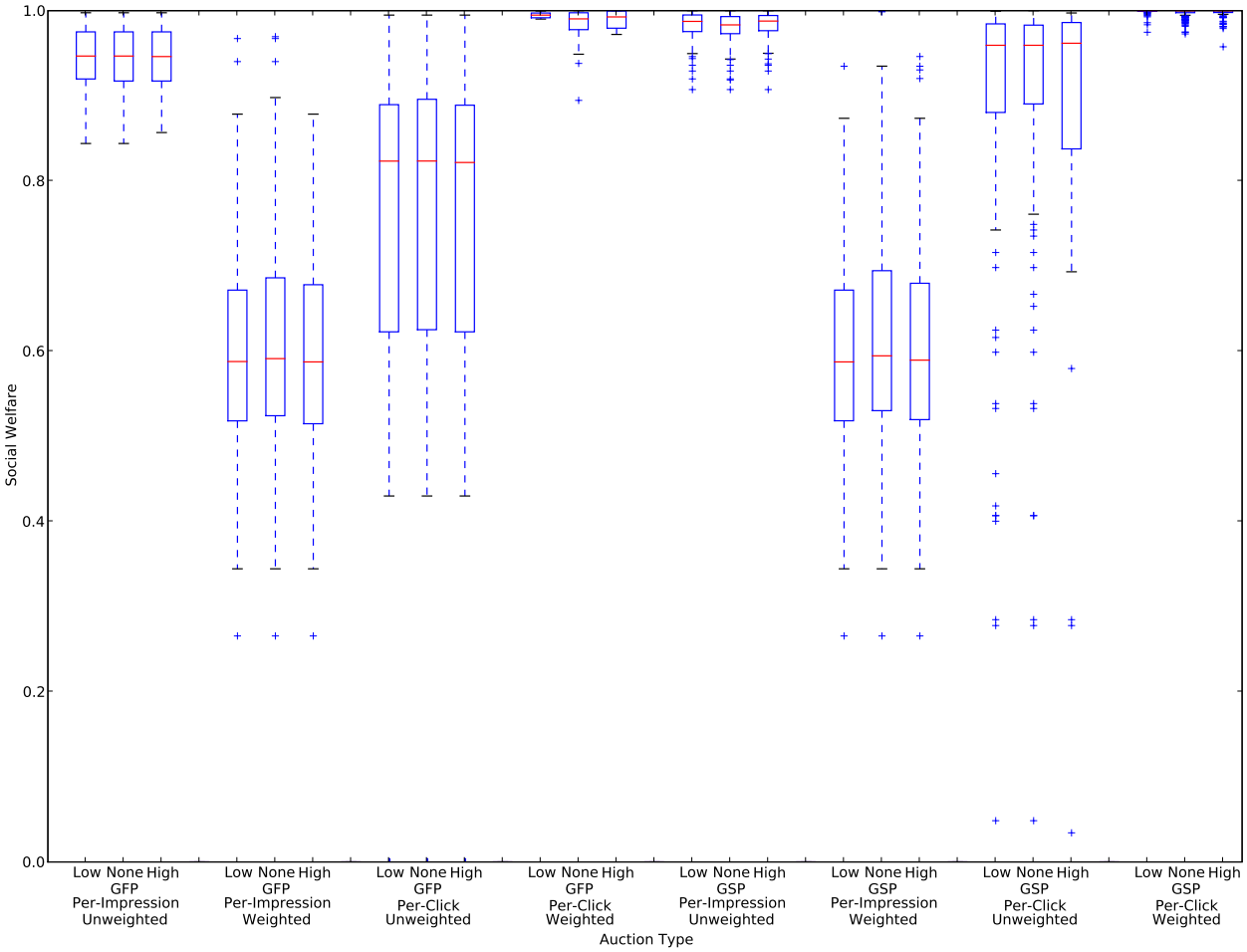

Figure 8: Expected social welfare in equilibrium under different auction types. The three columns for each auction type represent the different equilibrium biases: low bidding, no bias and high bidding from left to right.

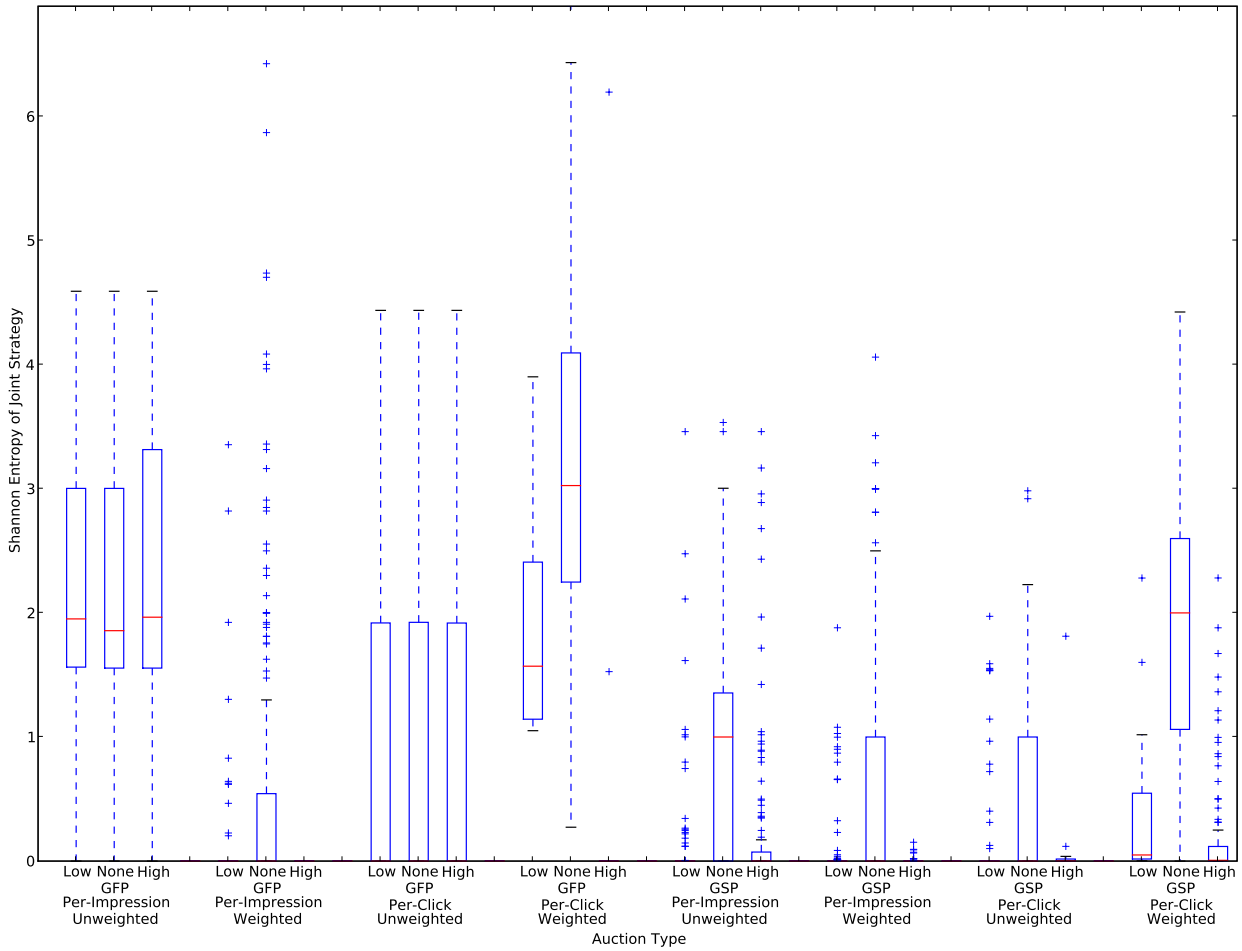

Figure 9: Entropy of the equilibrium under different auction types. The three columns for each auction type represent the different equilibrium biases: low bidding, no bias and high bidding from left to right.