Towards a Generic Coq Proof of the Truthfulness of Vickrey–Clarke–Groves Auctions for Search*

Pierre Jouvelot[1]
MINES ParisTech, PSL University, France
pierre.jouvelot@mines-paristech.fr

Lucas Massoni Sguerra
MINES ParisTech, PSL University, France
lucas.sguerra@mines-paristech.fr

Emilio J. Gallego Arias
Inria Paris, France
e@x80.org

Abstract

We present elements of a Coq/SSReflect proof of the truthfulness of the Vickrey-Clarke-Groves (VCG) auction algorithm for sponsored search (VCG for Search), variants of which are daily used by companies such as Google and Facebook for their advertising engines. We start from a formalization of the more general VCG mechanism, for which proving truthfulness, i.e., that bidders get the best utility by bidding their true value, is somewhat easy. We then show how VCG for Search can be seen as a functional instance of this mechanism, thus getting among other properties and for almost free a proof of a restricted version of the truthfulness of VCG for Search. Future work will focus on extending this preliminary result to the full theorem.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design

Keywords and phrases Formal verification, VCG auction, Sponsored search, Truthfulness

Acknowledgements We want to thank Tim Roughgarden (Columbia U., USA) for his great CS269I lecture notes and kind advice and Olivier Hermant (MINES ParisTech).

1 Introduction

Auctions for advertising space are a financial pillar of most internet-based sponsored search services such as Google. Each time a search request is performed by a user, interested digital publicity marketers automatically bid for the result-included web-page real estate dedicated to sponsored answers in order to promote their clients' products; this is done billions of times a day [6]. The correctness of the auction mechanisms implemented by these providers is thus of paramount concern, even more so when one envisions the possible future use of auctions in blockchain-based smart contracts, where code cannot be modified to correct bugs [1].

Getting formal assurance that auction algorithms are correct using proof assistants has been studied before (e.g., [3], [2], [4] or [5]). Our focus is on the Vickrey-Clark-Groves auction algorithm for sponsored search (VCG for Search), variants of which are heavily used in the industry [6]. We are also interested in studying how much the notion of instantiating mechanisms (see below) to algorithms can ease proof transfer, here for VCG for Search.

Using Coq/SSReflect, our contributions are (1) a specification of the VCG for Search algorithm, (2) a specification of the General VCG mechanism, together with proofs of three properties, namely no positive transfer, agent rationality and truthfulness (bidders get the

* MINES ParisTech, CRI, Technical Report A/748/CRI, March 2021.

[1] Corresponding author.

Listing 1 VCG for Search algorithm, assumming *bs* is sorted

```
Definition ctrs := k.-tuple ctr.
Definition bids := n.-tuple bid.

Variable (cs : ctrs).
Notation "'ctr_ s" := (tnth cs s) (at level 10).
Hypothesis sorted_ctrs : sorted_tuple cs.

Variable (bs : bids).
Notation "'bid_ j" := (tnth bs j) (at level 10).

Lemma slot_as_agent_p (s : slot) : s < n.
Definition slot_as_agent (s : slot) := Ordinal (slot_as_agent_p s).
Definition slot_pred (s : slot) : slot := ord_pred k s.

Definition externality (s : slot) :=
  let j := slot_as_agent s in
  'bid_j * ('ctr_(slot_pred s) - ('ctr_s)).

Definition price (i : A) :=
  if i < k then \sum_(s < k | i.+1 <= s) externality s else 0.
```

best utility by bidding their true value), and (3) a proof that VCG for Search is an instance of General VCG, which (4) helps translating these property proofs to this specialized case.

2 VCG for Search

In a VCG for Search auction, k slots, of type *slot*, have to be distributed among n bidders, or "agents", of type A, each of which providing a particular *bid*; one assumes that $k < n$. These slots typically correspond to a particular frame of a Web page, characterized by its statistical "click-through rate", in *ctr*, where the winning bidder's ad will be inserted. All these types are finite ordinals, e.g. `'I_n` for A, i.e., the sets of bounded natural numbers, here in $[0, n[$.

An auction is defined by two tuples, in *ctrs* and *bids*, indexed by slots and agents. The VCG for Search algorithm, given in Listing 1 (in the whole paper, Coq/SSReflect proofs are omitted, and some slight editing has been performed), expects thus as input a tuple *cs* of down-sorted rates and a tuple *bs* of bids, assumed as well to be down-sorted (see below). The agent i wins slot i (with thus $i < k$), paying for it $price(i)$ to offset the negative impact on the global social welfare incurred by her presence. This value, as proposed by Vickrey, Clarke and Groves, is the sum of all the externalities, i.e., financial losses, of the agents ranked after i according to *bs*, who thus do not get slot i.

For example, if $cs = (5, 3, 1)$ and $bs = (100, 50, 10, 4)$, then agent 0 will get slot 0 and pay $50 * (5 - 3) + 10 * (3 - 1) + 4 * 1 = 124$; agent 1, slot 1 for $10 * (3 - 1) + 4 * 1 = 24$; and agent 2, slot 2 for $4 * 1$ (agent 3 gets nothing; $cs[3]$ is assumed 0).

3 General VCG

VCG for Search is a particular instance of General VCG, an auction mechanism (see Section 4). For functional programmers, a "mechanism" is simply a higher-order function or module, here *VCG*. General VCG, in Listing 2, is abstracted over the type O of possible auction

Listing 2 General VCG mechanism

```
Variable (O : finType) (o0 : O) (i : A).

Definition bidding := {ffun O → nat}.
Definition biddings := n.-tuple bidding.

Variable (bs : biddings).
Local Notation "'bidding_ j" := (tnth bs j) (at level 10).

Implicit Types (o : O) (bs : biddings).

Definition bidSum o := \sum_(j < n) 'bidding_j o.
Definition bidSum_i o := \sum_(j < n | j != i) 'bidding_j o.

Definition oStar := [arg max_(o > o0) (bidSum o)].

Definition welfare_with_i := bidSum_i oStar.
Definition welfare_without_i := \max_o bidSum_i o.

Definition price := welfare_without_i - welfare_with_i.
```

outcomes, a particular instance *o0* (to ensure non-emptiness) and an agent i. Here, any agent, among n, is defined by its *bidding*, a finite function that values any possible outcome in the Coq domain *nat* of natural numbers. General VCG, given its last parameter, a tuple *bs* of *biddings*, must compute the outcome *oStar* that maximizes the total *bidSum* o of bids. In a truthful mechanism (see below), where the bids of agents and their "values" coincide, this outcome maximizes the global good, or "welfare". For agent i, the *price* she accordingly has to pay to win whatever is in *oStar* for her is a penality induced by the impact on the global good of her presence in the bidding process (*welfare_with_i*) compared to when she is not (*welfare_without_i*, which would have yielded a possibly different optimal outcome).

We formally prove that General VCG enjoys useful properties such as "no positive transfer" (all prices are positive, and thus the auctioneer does not have to pay bidders), rationality (for any agent, the price is less than the value of the outcome for him) and the most important one, truthfulness (see Listing 3). General VCG assumes the existence, for any agent i, of a valuation *value* i that he assigns to any outcome in O. The *utility* of the bidding result for i, among n agents bidding *bs*, is then the difference between whatever the perceived value is and the price paid (note the three explicit arguments to the mechanism functions *oStar* and *price*). The truthfulness property that Theorem *truthful* expresses is key. It states, that all things being equal, as stated by $differ\_only_i$, the only way i can increase its utility is by bidding, for any outcome o, what is for him its true *value* in o.

4 VCG for Search as a General VCG Instance

Formally showing that VCG for Search is an instance of General VCG requires constructively showing there exist values O, $o0$ and BS such that, for any agent i and bids bs, one can prove that the VCG for Search *price bs i* is equal to the General VCG *VCG.price O o0 i BS* (the prefix *VCG* shows that we put General VCG in a Coq module). We exhibit these proper definitions in Listing 4, where we introduce the *biddings* function that maps any tuple of bids *bs* to its appropriate General VCG version.

Listing 3 Truthfulness of General VCG (i is defined previously)

```
Variable (value : bidding O).

Definition utility bs := value (·oStar O o0 bs) - (·price O o0 i bs).

Definition differ_only_i bs' :=
    forall j, j != i -> tnth bs' j = 'bidding_j.

Theorem truthful bs' :
  'bidding_i =1 value ->
    differ_only_i bs' ->
      utility bs' <= utility bs.
```

Listing 4 VCG for Search parameters for General VCG

```
Notation "'bidders" := (k.-tuple A) (at level 10).

Structure O :=
   Outcome {obidders :> 'bidders;
            ouniq : uniq obidders}.

Variable (bs : bids).
Notation "'bid_ j" := (tnth bs j) (at level 10).
Hypothesis sorted_bs : sorted_bids bs.

Definition bid_in (j : A) (s : slot) := 'bid_j * 'ctr_s.
Definition t_bidding (j : A) (o : 'bidders) :=
    if j \in o then bid_in j (slot_of j o) else 0.
Definition bidding (j : A) := [ffun o : O => t_bidding j (obidders o)].
Definition biddings := [tuple bidding j | j < n].

Definition t_oStar := [tuple widen_ord le_k_n j | j < k].
Lemma oStar_uniq : uniq t_oStar.
Definition oStar := Outcome oStar_uniq.
```

A VCG for Search outcome, in O, is a k-tuple of agents that satisfies the *uniq* predicate, enforcing no repetition of agents. Note that a set wouldn't be appropriate here, since the order of agents matters for computing prices. For any *bs*, the corresponding *BS* is defined as *biddings bs*, an n-tuple of finite functions mapping any outcome o to a natural number. As seen in *t_bidding*, any agent j, if present in a given outcome o, bids in General VCG the value *'bid_j * 'ctr_ s*, where s is the slot number of j in o; otherwise, he bids 0. For the final parameter, *o0*, we can use *oStar*, which is the k-tuple that includes the highest k bidders. These are the winning ones according to VCG for Search, and we indeed prove that *oStar* maximizes the VCG for Search-specific global welfare.

5 Truthfulness of VCG for Search

The main advantage of showing that VCG for Search is an instance of General VCG is that we can reuse the formal proofs of the latter's properties to help prove the same for VCG for Search. We focus on truthfulness. Here an additional parameter, namely *value_par_click*, needs to be specified, taking into account that VCG for Search deals with per-click prices, while

Listing 5 Equivalence of utilities (*sOi* coerces agents to slots)

```
Section Utility.

Variable (bs0 : bids) (i i' : A) (iwins : relabelled_i_in_oStar i i' bs0).
Let bs := tsort bs0.

Definition click_rate := ('ctr_(sOi i'))%:Q.

Definition per_click (n : nat) := n%:Q / click_rate.

Definition price_per_click := per_click (relabelled_price bs0 i').

Definition utility_per_click :=
  (* max needed since VCG.utility is a nat. *)
  maxr ((value_per_click i)%:Q - price_per_click) 0.

Definition utility := utility_per_click * click_rate.

Definition vcg_utility (i : A) v bs := (VCG.utility o0 i v bs)%:Q.

Definition value_bidding :=
  [ffun o : O => (value_per_click i * 'ctr_(sOi i'))%nat].

Lemma eq_VCG_utility :
  0 < click_rate -> utility = vcg_utility i' value_bidding (biddings bs).

End Utility.
```

General VCG parameters we used up to here do not (we use rationals, in the ring Q). Listing 5 shows how *value_per_click* is combined with click rates to build the argument *value_bidding* passed to *VCG.utility*. We prove, in Lemma *eq_VCG_utility*, that *VCG.utility* is indeed equal to the VCG for Search-specific *utility*. The function *utility_per_click* uses the *max* function to force the utility to be positive, since we use natural numbers in the *VCG* module. Note that the lemma uses two additional conditions. The first one is *iwins*; it ensures that agent i is indeed a winner, meaning that its "relabelled self" i', after the required sorting down, via *tsort*, of the initial bids *bs0*, is indeed among the winners, the k first bidders, in *oStar*. And, since we are dealing with per-click utilities, the click rate must also be non-null.

The main lemma, *VCGforSearch_stable_truthful*, is stated in Listing 6. Two additional conditions are needed to prove the truthfulness of VCG for Search. The first one is similar to *iwins*, discussed previously, but applies when i bids differently, as expressed in $bs0'$. Note that here i is supposed, in both cases, to be relabelled as the same agent i', i.e., at the same position in the sorted bids, via the sorting process, thus limiting this lemma to "stable" changes of i's bid. The second condition, $uniq\_oStar'$, states that the only optimal outcome is *oStar*, which we conjecture is only true when all bids are distinct (when there are equal bids, the agents could be swapped). We discuss these two issues in Section 6.

The main advantage of the previous proof that VCG for Search is an instance of General VCG is that the proof of *VCGforSearch_stable_truthful* relies mainly on a Coq *apply* : *VCG.truthful* command.

Listing 6 Truthfulness of VCG for Search

```
Definition value_per_click_is_bid :=
  [forall o : O, per_click i' (bidding (tsort bs0) i' o) == (value_per_click i)%:Q].

Definition differ_only_i (bs bs' : bids) :=
  forall (j : A), j != i' → tnth bs' j = tnth bs j.

Lemma vcg_differ_only_i (bs1 bs2 : bids)
      (diffi : differ_only_i bs1 bs2) :
  VCG.differ_only_i i' (biddings bs1) (biddings bs2).

Lemma VCGforSearch_stable_truthful (bs0' : bids)
      (iwins' : relabelled_i_in_oStar i i' bs0')
      (uniq_oStar' : singleton (max_bidSum_spec (tsort bs0'))) :
  value_per_click_is_bid →
  differ_only_i bs (tsort bs0') →
  utility bs0' i i' <= utility bs0 i i'.
```

6 Future Work

This formalization lacks the full theorem regarding VCG for Search truthfulness, i.e., when i is not stable in $bs0'$. The expected constraint for this would be $relabelled\_i\_in\_oStar\ i\ i''\ bs0'$, for some proper i''. It is not yet clear how this can be obtained without digging into the specifics of the VCG for Search algorithm.

A couple of assumptions also remain in the current framework. The first assumes that all the outcomes that maximize the global welfare are equal, which is not true, since one could swap two agents with identical bids. A proof of the irrelevance of this choice would be warranted. A second one has to do with the simple sorting process of bids, *tsort*, and other tuples; a few properties related to this sorting process are presently assumed.

Finally, looking at other variants of VCG for Search could be interesting, since real-time auctions now include more advanced features than static click-through rates.

7 Conclusion

We describe on-going work that intends to provide a Coq/SSReflect formalization of the VCG for Search auction algorithm and of its properties, derived, as much as possible, from its instantiability from the General VCG mechanism. The whole project is open source and available at `https://github.com/jouvelot/VCG_Stable`.

References

1 Mouhamad Almakhour, Layth Sliman, Abed Ellatif Samhat, and Abdelhamid Mellouk. Verification of smart contracts: A survey. *Pervasive and Mobile Computing*, 67:101227, 2020. URL: `http://www.sciencedirect.com/science/article/pii/S1574119220300821`, `doi:https://doi.org/10.1016/j.pmcj.2020.101227`.

2 Wei Bai, Emmanuel M. Tadjouddine, and Yu Guo. Enabling automatic certification of online auctions. *Electronic Proceedings in Theoretical Computer Science*, 147:123–132, Apr 2014. URL: `http://dx.doi.org/10.4204/EPTCS.147.9`, `doi:10.4204/eptcs.147.9`.

3 Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. Computer-aided verification in mechanism design. *CoRR*, abs/1502.04052, 2015. URL: `http://arxiv.org/abs/1502.04052`, `arXiv:1502.04052`.

4 Marco B Caminati, Manfred Kerber, Christoph Lange, and Colin Rowat. Sound auction specification and implementation. Discussion papers, Department of Economics, University of Birmingham, 2015. URL: `https://EconPapers.repec.org/RePEc:bir:birmec:15-08`.

5 Manfred Kerber, Christoph Lange, Colin Rowat, and Wolfgang Windsteiger. Developing an Auction Theory Toolbox. In Manfred Kerber, Christoph Lange, and Colin Rowat, editors, *AISB 2013*, pages 1–4, 2013. proceedings available online. URL: `http://www.cs.bham.ac.uk/research/projects/formare/events/aisb2013/proceedings.php`.

6 Tim Roughgarden. *Twenty Lectures on Algorithmic Game Theory*. Cambridge University Press, 2016. `doi:10.1017/CBO9781316779309`.