Abstract

: Thanks to Glen Weyl's excellent feedback on my original post regarding SGD in quadratic funding, which had significant challenges with the complexity of the multi-goods case, I've significantly refined the framework. I argue that the bid language suggested in the original post is still a substantial improvement over the current bid language, present a marginally even more expressive bid language as well as a framework for "compiled languages". I close with open questions and desiderata for the further design of QF query and bid process.

Recapping Revelation

The basic form of the revelation principle is that you can hand your preferences to the mechanism designer and the mechanism designer will play your strategy for you. The limitation of this basic form is that it may ask for too much information (the classic example is a combinatorial auction, in which there are 2^m

combinations of m

goods; in some sense, the combinatorial public good problem is a harder variant of combinatorial auction, i.e. it has real-valued assignments and a free-rider contingency). Hence, indirect mechanisms make limited queries

about your preferences (which may not necessarily return samples but may instead return statistics) to attempt to model and optimize over preferences under size and compute constraints. The relevance of the relevation principle is now that it is incentive-compatible to cooperate with the model.

It's worth noting that a strategic choice is

a revelation; hence, technically, every mechanism is based on revelation. Inter-participant revelation may be analogized to the uncertainty which comes from the "firmness" of the model.

Quadratic funding is applicable to settings with varying levels of challenges. The lowball in terms of revelation might be consortiums of companies - the participants and options are manageable in number, the decisions are of high consequence per participant, and each participant has many hours to devote to the process. Group houses and communes are another smaller-scale testing environment. However, municipality-scale public goods consumed directly by individuals starts to show challenges (it is likely that some of these are insurmountable and would require hierarchies, i.e. smaller municipalities representing within larger municipalities). When there are both a large number of users and a large number of public goods, even showing every user every public good

is already potentially infeasible.

The Model

The system has a model M = V(i, p, F_1, …, F_P)

of size S

of the users' private valuations of goods (technically, the model only needs to compute sufficient statistics for optimization, but we proceed without this detail). Each round we perform a question-and-answer conversation, i.e. given a query q_i

, the user optionally responds with answer a_i

and/or a peer review

r_i

(which may want/need to be subsidized), which improves queries and query processing for later users, as well as helping the sponsor adjust productivity priors (this is a tangential idea and the governance problem implied is outside our current scope). The model M

updates to M' = V(i, p, F_1, …, F_P, a_1, …, a_i)

, with a compute/size update cost

u(M, a_i)

, then recomputes the best strategy equilibrium (this does not have to be done for every a_i

), incurring a computational solver cost

of o(M')

, and making the resulting funding amounts publicly available (this allows users to update their answers, at the cost of some differential privacy).

The model loss is the difference between realized and optimal welfare, and a query loss could be defined as L(q_i) = p(q_i) + d(q_i, a_i, r_i)

where p

is the cost of reading and processing the query q_i

and d

is the discursive cost of producing a_i, r_i

.

Note that neither d

nor p

nor L

strictly conform to classical notions of “communication complexity”. We say q_i

is expressed in query language

, a_i

is expressed in bid language

, and r_i

is expressed in review language

. We can roughly expect the user to process queries until there is not enough improvement in private benefit to justify the cost.

Example of desired outcome:

An R&D team needs up to $2m to pursue a new project. They submit this project proposal, with milestones & deliverables, to a QF round of 2000 people, of which 25 stakeholders receive 80% of the benefit. The platform discovers these connections and facilitates group deliberation which ends up costing each stakeholder $1000 to come up with a valuation model in the context of other available options. Most of the 975 other participants are not contacted, although some misrecommendations are made and a few peoples’ time is wasted.

Choosing a Query Language and Bid Language

We may optimize the bid language for expressivity, word length, ability of the model to process, and/or intuitiveness. We assume that adding more words to the bid language does not directly impose a cost to the user but may impose costs in terms of displacing other words, or being hard for the model to process.

Current QF

The current QF language works as follows: the model runs a search/ranking/recommendation engine to retrieve a set of projects P

, queries its state for the funding and match amounts F_p, M_p

for each project, then queries the user with the query (P, F, M

, “how much do you want to contribute?”). The user replies with a set of contributions (p, c^p)

and/or a next search query. The model parameters are c_i^p

for all participants i

and projects p

, where each parameter starts at 0

and is updated to c^p

upon an answer from participant i

. Note that there are N*P

parameters for N

participants and P

projects, but only \mathcal{O}(k)

nonzero parameters in terms of total response length k

; hence, it is advantageous to use sparse representation and sparse math.

The query answer reveals the user's marginal private benefit to be c^p/(c^p + M_p)

at the shown funding amount F_p

, and the model imputes a marginal private benefit of c^p/(c^p + M_p')

at the new

funding and match amounts M_p', F_p'

, so that the model returns the untouched c^p

values.

There is significant loss in a few ways: 1) the model imputes from arbitrarily stale data, and 2) the model is "forgetful" i.e. it only remembers the latest query answer from each user. As we showed in our previous post, it may take many rounds to converge (played out over time, imposing significant context-switching costs; broadly, it is generally not viable to have the user log in very frequently), we may not reach convergence at all, and the convergence process may even be sabotaged.

Indeed, 1) implies that "spoofing" is possible in dynamic CQF (Buterin, Weyl, Hitzig 2018), i.e. a user may intentionally "pump" displayed match amounts. Preventing this could be attempted with activity rules (CQF introduces a tension to activity rules, which is that we'd like to give a "firm offer", but CQF participants should be unwilling to give firm offers since their fill may be affected by other

goods; expanded bid languages may be more amenable to activity rules), but penalizing this would require charging users in intermediate rounds based on their reported contribution, which would change the mechanism (this is nonetheless a possible research direction). We may also consider hiring expert reviewers to be privileged "first movers" who provide firm offers and peer review in order to aid the discovery process (the cold start problem is twofold in that not only will an unpopular project not show up in my feed, but I will be unmotivated to evaluate it even if it does, due to the low displayed match).

Honest users can try to play around the model loss, e.g. by discounting the data displayed; however, this places an undue information burden on the user.

We see opportunities to improve the model without changing the bid language

, via more intelligent imputation. For example, if I increase my contribution in response to updated funding numbers, which later revert, then the mechanism could also roughly revert my contribution (this could defend against spoofing somewhat). One caution is that some "smart contribution adjustment" model improvements may cause strange/risky-seeming behaviour, which imposes a discursive cost (we could mitigate this by making it an opt-in feature that only needs to be turned on once by the user).

There is likely a limit to how much we can improve the model subject to the limitations of this bid language. We seek to productively expand the bid language at minimal cost.

Complementarity-Naive Contingent QF (CNCQF)

In our previous post, we suggested that users should also be able to express contingent valuations of a good. To limit the complexity, we only accept valuations contingent on the funding level of the good and not of other goods (complementarities), i.e. (V_p, F_p, c^p)

. The query language is untouched save for an additional string "(optional) add contingent contribution", and the bid language is a strict superset, as if we receive an answer (p, c^p)

which does not specify F_p

, then we assume F_p

is the displayed funding level, and we can even use the same imputation as original QF. This ensures that there are no unforced query losses. With regards to the increase in model cost, see the previous post (again, sparse math will help here). The model is "naive Bayes" in the sense that we let V(i, p, F_1, …, F_P) = V(i, p, F_p)

which is interpolated from the user answers.

When/why would a user provide a contingent valuation? A few reasons:

- It is cheap to do so if the user is already on the page and thinking about their non-contingent valuation
- It protects the user from important contingencies never being hit, such as"cold start" goods, as well as allows the user to bid on projects that have not hit an explicitly set funding minimum
- It protects the user from spoofing
- The user may have given an overall budget to the system, which is enough to fund all the projects the want to; hence they must fund contingently
- the broader picture of the above points is that it offers downside/upside protection for ~free
- It is cumbersome to have to log in again later, even if waiting results in needing less responses overall

We may compare providing contingent valuations to "providing liquidity" in a market, which is costly but still often desirable due to the desirability of fills and the reality of stepping away from the computer.

We hence expect that this option would actually be used frequently to great benefit, and if not used would impose no cost.

If our mechanism is multi-round, we may force users who've provided contingent valuations to commit to some threshold (we need more than one interpolation point to generate any reasonable "commitment frontier"). We may start with a live visualization of the user's implied contingent valuation given a standard QF bid and, if the user decides to modify a point, show how the valuation curve as well as the commitment curve evolve. We may also give suggestions as well as descriptions of deliverables at sampled funding levels (these may be provided by the project who is fundraising, or peer review).

Complementarities

Should we add language to express complementarities, or allow complementarities to be revealed through multiple rounds of CNQF? One challenge is that there is a significant curse of dimensionality in sampling vectors of funding amounts. A possible dimension reduction (call this UCCQF for Uniform Complementarity Contingent QF) could be to add words of the form "penalize the grouping of goods {p_1, p_2, … p_n}

by $k" which causes the model to add a penalty of k

times the geometric mean of F_{p_i}

. This has the appealing property that it does not mess with the model's interpolation; instead it adds \mathcal{O}(n)

operations to the gradient computation. There is also a natural language interpretation of this bid-word, which is that $D of investment into each of the goods creates $kD less private benefit for me than the sum of $D of investment into each good alone.

It's unclear whether most people would have the skills to use these words directly (although note that if someone learns a bid language once, they can use it in any instance of any mechanism with the same bid language), although they again do not compromise the existing bid language, only adding an "advanced optional feature".

Another requirement for eliciting words of this type is that the UI surfaces potential complementarities (which may be learned from bid data of all users; this may be an open research topic).

We later also explore instead the possibility of "compiled languages" which compile to the "lower-level" UCCQF language or other "machine language".

Interdependent Values

Our valuations may also depend on other people's

valuations. Indeed, if we trust certain friends or experts, we can save a significant amount of time on evaluation and simply submit words of the form "unilaterally give Y% weight to person X's valuation" or "give Y% weight to person X's valuation of project Z". Each word of this type would add \mathcal{O}(1)

operations to the gradient computation. We'll call this UCCQF+.

A UI for recommending these bids could consist of showing a visualization of bids of a few recommended peers.

Another client-side way for users to defer to other peoples' judgments would be an option to snooze a recommendation until the good reaches a certain level of funding.

Natural Language

Glen Weyl speculates "Some of the best reseach directions will, I think, be much different from this, involving people expressing in natural language values and working over time to work out the trade offs they face with a variety of XR powered data vis."

We offer an interpretation of this. "Natural language" may be fed directly into the model, or may be first compiled into intermediate "machine-readable bids". We argue that the latter setup is more promising, since 1) it is easier to design the server-side model, 2) the model processes more easily, and 3) it allows the user to iterate quickly on the bid before hitting the model.

One may run a conversational model locally which intermediates between the user and UCCQF+ and submits UCCQF+ -language bids on the user's behalf, but not without showing the user a live-updating interactive visualization of the bids / resulting model update. The conversational model would help the user discover relevant goods and contingencies as well as automatically generate machine-readable bids (ideally, generating a large amount of informative bid per round with little/smooth work by focusing queries on known unknowns).

Empathy

Another benefit of good dialogue is that it increases empathy, such that participants' preferences shift to also represent non-participants' preferences - since e.g. foreigners or future citizens may not be able to participate in the mechanism. We may also attempt to represent the preferences of other participants whose preferences are too small to bother transacting with the mechanism. However, there is a coordination issue of who's responsibility it is to care about whoand a clear division-of-emotional-labor system could be of great value. Also, this is clearly outside the realm of incentive-compatibility, but still worth thinking about.

A better design principle for the "preferences too small" case may be to lump smaller public goods into larger bundles (in addition to of course, expert reviewers fronting investigation costs).

Conclusion

We have presented a taxonomy and framework for evaluating indirect implementations of quadratic funding. We suggest that expanding the bid language to be able to express valuation contingencies on funding amounts, a limited amount of complementarity, and interdependent values solves several issues with the current bid language. However, we also acknowledge that there is significant further work to be done in this field. A major open question is whether natural language conversation models may be used to generate contingencies in the proposed "machine language".

Research Directions

- developing an appropriate "privileged first mover" mechanism
- optimizing the information flow of early peer review
- activity rules and penalties for spoofing
- "smart contribution adjustments" within the original bid language
- improvements on model loss and computational efficiency
- further work on "machine bidding language" which is expressive and efficiently processable
- conversational models which generate aforementioned machine language
- ability to visualize/inspect machine language bids as well as model state
- explore vs exploit trade-off in querying users - likely that having priors about users' expertise and interests would assist here
- coordination of empathy
- privacy

Related Reading

- Noan Nisan, "Bidding Languages" (Is there an equivalent to "OR/XOR formulae bids" in the context of QF?)
- Teytelboym et al, "Discovering Auctions: Contributions of Paul Milgrom and Robert Wilson"