Response to Prof. Milgrom and Prof. Ausubel's *Comments on the Second Wye River Package Bidding Conference*

Kevin Leyton-Brown
kevinlb@cs.stanford.edu

In this document I respond to three sections of Prof. Milgrom and Prof. Ausubel's analysis in their *Comments on the Second Wye River Package Bidding Conference*. First I examine the expressivity of the auction #31 bidding language. Second I discuss some computational issues that arise from the choice of bidding language, concentrating on computational benefits of the OR-of-XOR language and on how additional features required for straightforward bidding could be added to the OR-of-XOR language without serious computational consequences. Finally I argue that it should not be computationally necessary for the auctioneer to drop bids from one round to the next. I quote from Prof. Milgrom and Prof. Ausubel directly in each case to put my comments in context.

1 Auction #31 Bidding Language Expressivity

Prof. Milgrom and Prof. Ausubel wrote:

> As for the bidder interface, the computer scientists conception differentiates among alternative package designs partly on the basis on the "expressiveness" of the "bidding language," that is, the ability of bidders to make bids that accurately reflect their own package preferences or values. The auction #31 interface necessarily connects bids within each round using the logical disjunction "OR." This means that if a bidder makes, say, two bids in a round, the auctioneer can accept one or the other or BOTH. A problem with these rules is that this OR language fails to be "fully expressive." A bidder who wants to acquire either A or B but NOT both cannot express this preference with a bid made in a single round. The FCC mitigated that problem for auction #31 by adopting Pauls suggestion that bids in different rounds should be treated as mutually exclusive alternatives (called an "XOR" relationship), while allowing a bidder who does not want bids to be mutually exclusive to renew them in the current round.

> The actual auction #31 rules thus involves a hybrid of OR and XOR expressions that seems to have confused some commentators. Focusing on the bidding rules within a single round apparently led some to conceive of auction #31 as essentially using the "OR" language. If only OR bids were permitted, withdrawals would be needed to allow a bidder who has bid on A and now wants to bid on B to avoid the exposure problem. In the actual auction #31 rules, however, the mutual exclusivity of bids across rounds eliminates the inter-round exposure problem and with it the need to allow bid withdrawals as protection.

When bids expire after two rounds—as I understand that they do under the current auction #31 rules—then these rules do not induce a fully expressive bidding language even when bids *across* rounds are considered. A consequence is that the inter-round exposure problem cannot be completely overcome under the current rules. In this case the auction implements a bidding language of the form:

$$\{bid \text{ OR } bid \text{ OR } \ldots \text{ OR } bid\} \text{ XOR } \{bid \text{ OR } bid \text{ OR } \ldots \text{ OR } bid\}.$$

This is certainly more expressive than the OR language. For example a bidder who valued A XOR B, as in Prof. Milgrom and Prof. Ausubel's example, would be able to express this valuation in the auction #31 language but not in the OR language (ignoring differences in the bidder's treatment under the auction rules for placing a set of bids across two rounds rather than in a single round). Indeed, as Prof. Milgrom and Prof. Ausubel point out, the fact that bidders may renew the same bid from one round to another means that the auction #31 language is strictly more expressive than the OR language. However, the auction #31 language can still not represent the valuation of a bidder who wants A XOR B XOR C as it allows only two sets of bids to be mutually exclusive. Indeed, auction #31 bids are not fully expressive as long as they expire after any number n of rounds, because a set of $n + 1$ XOR'ed bids could not be expressed. Likewise, even without bid expiration, a bidder cannot express the desire for $n + 1$ bids to be mutually exclusive in the n^{th} round of the auction. Finally, a bidder cannot update the price offer on more than one bid per round without causing the side effect that the two updated bids may be jointly satisfied. It is only when bids never expire, bidders get an arbitrarily large number of rounds in which to place XOR bids, and bidders are guaranteed not to update more than one bid per round that we get the XOR-of-OR bidding language, which is a superset of the XOR language and therefore fully expressive.

2 Computational Issues in Bidding Languages

Prof. Milgrom and Prof. Ausubel wrote:

> The OR-of-XOR structure is said to have two advantages over the XOR structure. First, to the extent that bidders wish to place bids with a corresponding additive structure, it enables bidders to do so easily and compactly. However, straightforward bids in an ascending package auction do not generally have such an additive structure. The goal should be to create a simple interface that makes straightforward bidding easy. This requirement is orthogonal to the OR-of-XOR structural issues.
>
> The second advantage of OR-of-XOR is that reporting bids in this form simplifies the winner determination problem. This is useful, but secondary. Still, it is encouraging that computer scientist Kevin Leyton-Brown reported that it would be possible to keep these computational advantages of an OR-of-XOR interface even with the changes required to make straightforward bidding easy.

I will respond to two issues here. First, I'll elaborate upon what is meant by the claim that the OR-of-XOR's bidding language improves computation. Second, I'll go through some of the bidding language modifications that would support straightforward bidding, and discuss their computational consequences.

2.1 Computational Advantages of OR-of-XOR

As long as bidders are allowed to place fully expressive OR-of-XOR bids, they are also allowed to place simple XOR bids.[1] This means that there is no computational benefit to an OR-of-XOR bidding language over a simple XOR bidding language if bidders' bids never have an additive structure, as bidders will place simple XOR bids in both cases. However, even in this degenerate case, there is no additional computational cost for offering an OR-of-XOR bidding language. In practise there will be computational savings if the OR-of-XOR language is ever used by bidders; furthermore bidders will have access to a more compact and intuitive way of specifying their valuations when their valuations include additivity.

To see where this computational savings comes from, and to see why there *is* a computational advantage to using either an OR-of-XOR or simple XOR language instead of the XOR-of-OR language regardless of whether bids have an additive structure, it is useful to consider how the bidding language interacts with the optimization algorithm. Intuitively, most ways of setting up the optimization problem do not give the algorithm information about which bidders placed which bids; instead, the algorithm considers the set of all bids to be independently satisfiable (thus, OR'ed). This works for bids that originated from

[1] When bidders submit only a single XOR'ed set, OR-of-XOR bids are equivalent to simple XOR bids.

different bidders, but as just stated does not allow bidders to specify mutual exclusivity between their bids. A solution that Yoav Shoham and I proposed is to introduce "dummy goods" into bids: goods that are not actually sold in the auction, but which are added to all bids in an XOR set so that the optimization algorithm cannot allocate more than one of them.[2]

If we consider the process of "expanding out" bids from the bidding language into OR bids with dummy goods, we see that simple XOR bids will have a single unique dummy good added to all bids by the same bidder. We can also represent OR-of-XOR bids with a single dummy good, but we have to create more bids: essentially we have to ignore the additivity between the bidder's valuations for the XOR'ed sets and make a single bid for every set of bids that could be allocated to the bidder. In the worst case if there are m sets and each set i contains n_i goods, the number of bids we will have after the expansion is:

$$\prod_{i=1}^{m}\Big(2^{n_i}-1\Big).$$

This is terrible: the number of expanded bids grows exponentially with the size of the largest set and geometrically with the number of sets. For example, two sets of ten bids each can translate into more than a million bids! Luckily there is a way of expanding out the OR-of-XOR language that does not increase the number of bids at all: we can use a unique dummy good for each XOR'ed set rather than for each bidder. This approach requires slightly more dummy goods (equal to the total number of sets rather than the number of bidders) but reduces the number of bids by exploiting the additive structure in the bidder's valuation. Essentially, we do not have to explicitly expand out all the full bundles in which the bidder is interested—instead, we can use the optimization algorithm's OR language to represent these sets implicitly. Continuing the previous example, two sets of ten bids each could be represented with twenty bids and two dummy goods. Since this second expansion technique creates exponentially-fewer bids in the worst case, it can lead to much shorter running times for the optimization algorithm.

Unfortunately, things do not work so well when expanding out bids that were expressed in the XOR-of-OR language. As long as there is more than one XOR set (in the case of auction #31, as long as a bidder has placed any bids in each of two consecutive rounds) we have no alternative to expanding out the bids as above, making a new bid for every set of old bids that could be allocated to each bidder. As above, the expansion generates $\prod_{i=1}^{m}(2^{n_i}-1)$ bids. This illustrates why the use of XOR's is expensive in auction #31: increasing the number of sets causes a geometric increase in the number of bids, which has the potential to significantly slow down the optimization algorithm.

[2] Alternate techniques exist, but I stick here to the one I'm most familiar with. The same issues I describe would come up under other schemes too, because dummy goods are essentially a way of thinking about the addition of new constraints between bids, and the number of bids once dummy goods have been added is a way of thinking about the size of the search space given the new constraints.

In the worst case the OR-of-XOR bidding language does not yield computational benefits, because bidders' valuations may not have any additive structure for this language to exploit. Likewise, bidders using the XOR-of-OR language can also place simple XOR bids by submitting OR sets of one bid each. In this worst case, there is no computational difference between using the two languages. However, there are types of structure in bidders' valuations that each bidding language can exploit. The important difference between the languages is that when structure *does* exist, the OR-of-XOR language leverages this structure to achieve better computational performance with search algorithms such as CPLEX or Prof. Sandholm's BidTree, whereas the XOR-of-OR language does not.

2.2 Additional bidding language features

Here I'll go through some additional bidding language features that are important for real FCC auctions, and discuss how they can be incorporated into the OR-of-XOR bidding language.

2.2.1 A fixed amount to be subtracted from the price offer regardless of the number of bids that win

Consider a situation where a bidder wants to indicate that he will pay \$10M for A OR \$11M for B OR \$12M for C, but that it will cost him an additional \$2M to participate in the market at all. If this bidder had to use the standard OR-of-XOR bidding language he would have no choice but to submit a single XOR set containing all 7 bundles of interest to him with the appropriate price offers (e.g., \$8 for A; \$19 for AB, etc.). As above, the size of this set is exponential in the number of original OR bids. However, it is possible to add such constraints to the OR-of-XOR bidding language in a way that captures the additive structure in this bidder's valuation, enabling him to place only the three OR bids and the fixed amount. We can create constraints:

$$\forall i \, x' - x_i \geq 0$$

where x_i is an indicator variable for a bids by this bidder, and x' is a new indicator variable. In the objective function, we add the bidder's fixed cost of participation times x' (since the fixed cost is a negative number). This feature would require changes to the user interface, but it should not have a significant computational impact as all constraints are 0-1.

Although the fixed-cost-of-participation example might seem somewhat contrived, it turns out that the feature described above has an application to the support of straightforward bidding (as described by Milgrom and Ausubel). Straightforward bidders need to be able to increment all bids from the previous round by the same amount. Under a simple XOR bidding language this is trivial: all bid amounts may simply be adjusted. It is less obvious to know what to do under a bidding language that includes OR, as it is possible for more than one bid by each bidder to win. The fixed cost of participation solves this

problem: in the first round the fixed cost is set to zero (or to a positive number if the bidder really does have a fixed cost of entering the market) and then each round the fixed cost is incremented by the bid increment amount.

2.2.2 Budget limits

It is possible to envisage two different sorts of budget limits that bidders could place.

1. Bidders are allowed to specify the total amount that they would be willing to pay for any set of bundles as a single budget limit. They are restricted from placing single bids exceeding this amount (as these bids trivially violate the budget constraint) but they can place OR bids that could lead to allocations exceeding the limit. The optimization algorithm adds constraints ensuring that a bidder can never be allocated bids exceeding his budget limit. For example, if A, B and C are bids for \$10M, and the bidder has a \$25M budget limit, the bid A OR B OR C would be allowed, but at most two of the bids would be allowed to win.

2. Bidders may place budget limits on particular bundles. Continuing the above example, the bidder could indicate that the budget limit is \$30M if he is allocated A, and \$10M if he is not. This sort of budget limit may be questionable as in a sense it is another kind of bidding—in the extreme case where a bidder gives a budget limit for every bundle, there is no difference between specifying budget limits and bidding. In any case, I mention it here because it was discussed at the conference, and because the use of this sort of budget constraint is computationally feasible.

There are at least two ways that budget constraints could be encoded in an integer programming formulation:

1. For each bidder, the sum of the bidder's bid amounts times the relevant indicator variables is constrained to be less than the bidder's budget limit. This requires only a single IP constraint for each budget constraint. However, the constraint does not use 0-1 coefficients, and so the addition of such constraints may have a significant computational effect.

2. An alternative that does use 0-1 coefficients is to preprocess the bidder's bids to find all minimal satisfiable sets of bids that violate a budget constraint. For each minimal satisfiable set the sum of the corresponding indicator variables is constrained to be less than the number of variables in the constraint. For example, if x_1, x_{10}, x_{13} and x_{14} are indicator variables for bids by the same bidder which are jointly satisfiable according to the rules of the bidding language, these bids have a combined price offer that violates a budget constraint, and no subset of these bids have a combined price offer that violates a budget constraint, then the constraint is $x_1 + x_{10} + x_{13} + x_{14} < 3$. The construction of these constraints

is not computationally trivial (it requires time exponential in the number of bids placed by a single bidder) but it may be done as a preprocessing step, before other bidders have finished entering their bids. It is also worth mentioning that this approach has the potential to create a large number of constraints.

The use of budget constraints does affect computation, but the effect is not worsened by the use of the OR-of-XOR bidding language. In fact, if the use of this language leads to a smaller number of expanded bids then the budget constraints should also have a correspondingly smaller computational impact.

2.2.3 Bids that contradict earlier OR constraints

In a multi-round auction where bids cannot be withdrawn, bidders have a disincentive to split their bids into different XOR sets even when additivity exists in their valuations: they might decide later in the auction to place a bid that is XOR with bids in more than one set. For example, a bidder may have bid:

$$\{A \text{ XOR } B \text{ XOR } C\} \text{ OR } \{D \text{ XOR } E\} \text{ OR } \{F \text{ XOR } G\}$$

but could later decide to place the bid ABD, while still wanting it to be XOR with C and E. Since users have no economic disincentive for placing all their bids in a single XOR set, there is the danger that bidders would choose this strategy. (Note, however, that there is a *practical* disincentive for placing all bids in a single XOR set: it can require bidders to specify exponentially more bids!) The disincentive for OR bidding can be removed by allowing bidders to "merge" multiple XOR sets from a previous round, performing the exponential expansion described in the previous section, and allowing bidders to add new XOR bids to these sets. Thus bidders can end up with the same set of bids that they would have had if they had used the simple XOR bidding language all along, while retaining the computational and practical advantages of OR-of-XOR bidding in early rounds. Continuing the example, the bidder could merge the first two sets, so that his bid becomes:

$$\{A \text{ XOR } AD \text{ XOR } AE \text{ XOR } B \text{ XOR } BD \text{ XOR } BE \\ \text{XOR } C \text{ XOR } CD \text{ XOR } CE \text{ XOR } D \text{ XOR } E\} \text{ OR } \{F \text{ XOR } G\}$$

He could then add the bid for ABD to the first set. Note that the last set remains OR'ed, since it does not need to be merged in order for the new bid to be placed. Thus some compactness in the representation can be preserved even after sets are merged. As an aside, it might be desirable to maintain the OR-of-XOR representation internally after merging, until the bidder places a bid that requires that the sets be joined. This could of course be hidden from the user so that sets appear to merge even if new bids are not placed.

It is no surprise that merging sets has negative computational effects: it causes a blowup in the number of bids. However, there is another way of thinking about the computational cost of this feature: if users would otherwise submit

only single XOR sets, set merging may be seen as advantageous from a computational point of view because it removes the disincentive for users to submit OR-of-XOR bids.

3 Retaining a subset of bids from round to round

Prof. Milgrom and Prof. Ausubel wrote:

> If voluntary bid withdrawals are eliminated, the auctioneer may still wish, at its own initiative, to prune the bid list to simplify the computations during the auction. (We are agnostic about the need for such pruning, limiting our recommendation to how to proceed if pruning is to be implemented.) In this case, the bid quality index serves a second purpose: to guide the auctioneers decision about which bids to retain from round to round. One possibility is to retain each bidders N best bids, defined as any provisionally winning bids plus the highest quality bids among the others. In that way, bids with which other bidders might usefully combine are less likely to be prematurely deleted from the auction. The retained bids might also automatically include the most recent bids, in order to allow other bidders time to combine with new bids that are submitted.

Indeed, I think that Prof. Milgrom and Prof. Ausubel's agnosticism is warranted here: there are four reasons why I think that the auctioneer should not drop bids between rounds.

1. As Prof. Sandholm described in his talk at the Wye conference, the worst-case complexity of winner determination is polynomial in the number of bids. This means that the number of bids is not critical to the performance of the optimization algorithm, although of course worst-case runtime for optimization increases with the number of bids. Instead, the critical variable is the number of goods, which is decided by the auctioneer before the auction begins. This challenges the premise that pruning bids should ever be "necessary" for computational reasons.

2. If "low quality" bids are dropped then it is unlikely that the computational difficulty of the problem would be substantially changed. An optimization algorithm can usually remove uncompetitive bids from consideration with little computational effort, while the hardness of the problem is primarily determined by the more competitive bids.

3. There is no known technique for reliably predicting whether a given winner determination problem will be easy or hard.[3] Larger problems are harder on average, but it is not true that every large instance is harder than every

[3] This statement is no longer strictly true: since I wrote the first version of this document our group at Stanford, in collaboration with another group at Cornell, has been striving to develop techniques to address this problem. Though we've had some success at predicting

smaller instance. For example, if a very high bid is added to an existing set of bids, it might make the problem easier to solve because it is easy to prove that the high bid should be allocated, quickly eliminating many other bids. Thus, even if it *were* sometimes necessary for the auctioneer to drop bids, it might not be possible for the auctioneer to know this without solving the optimization problem.

4. I suspect that bid dropping from round to round would change economic incentives for bidders. I believe that computer scientists should do everything possible to avoid sacrificing economic desiderata for computational reasons; in this case, the computational problem is not overwhelming, and so dropping bids between rounds is probably not necessary.

whether winner determination problems will be easy or hard, our results are still preliminary. We are still a long way from understanding the empirical hardness of winner determination instances well enough for the FCC to rely upon our predictions in a real auction.