Specification for Implementing the Product-Mix Auction Solver

August 28, 2019

1 Definitions

1.1 Supply

A *good* is represented by an integer $j \in \{1, \ldots, N\}$.

A *supply curve* for good j is a step function σ_j, represented by a list of step widths $s_j^q \in \mathbb{R}^+$ and heights $\mu_j^q \in \mathbb{N}$.[1]

The supply curve should be nondecreasing, i.e. $\mu_j^{q'} \geq \mu_j^q$ for all $q' > q$.

A *supply* σ consists of:

- a set of goods $\{1, \ldots, N\}$;
- a supply curve σ_j for each good;
- a partial order $\preceq$ on the set of goods.

Let Covers(j) denote the set of goods that cover a good j in the partial order, i.e. its immediate successors.

A *base good* is a good that does not cover any other goods.

A *vertical* supply is one where $1 \preceq 2 \preceq \ldots \preceq N$, so 1 is the only base good, Covers(j) $= \{j+1\}$ for $j < N$, and Covers(N) $= \emptyset$. (The goods are ordered "vertically" in a single column.)

A *horizontal* supply is one where the partial order is trivial, i.e. every good is a base good and Covers(j) $= \emptyset$ for all j. (The goods are ordered "horizontally" in a single row, with no good above any other.)

The *size* of supply curve σ_j is the sum of its step lengths:

$$\text{size}(\sigma_j) = \sum_q s_j^q$$

[1] That is, $\sigma_j(x) = \sum_q \mu_j^q \chi_j^q(x)$, where χ_j^q is the indicator function for the q^{th} step and e_j^q is the left endpoint of the step, given by

$$\chi_j^q(x) = \begin{cases} 1, \text{ if } e_j^q < x \leq e_j^q + s_j^q \\ 0, \text{ otherwise} \end{cases} \qquad e_j^q = \sum_{1 \leq k \leq q-1} s_j^k.$$

The *size* of a supply σ is the sum of the sizes of the supply curves for the base goods:

$$\text{size}(\sigma) = \sum_{j \text{ base good}} \sum_{q} s_j^q$$

(thus $\text{size}(\sigma) = \sum_q s_1^q$ if σ is a vertical supply and $\text{size}(\sigma) = \sum_{j,q} s_j^q$ if σ is a horizontal supply).

1.2 Bids

A *bid* consists of:

- a *bid label* $i \in \{1, ..., M\}$;
- a *total quantity* $k_i \in \mathbb{R}^+$;
- for each good j:
  - a *bid value* $v_j^i \in \mathbb{N}$;
  - a *maximum quantity* $\kappa_j^i \in [0, k_i]$;
  - a *trade-off coefficient* $a_j^i \in \mathbb{R}^+$.

An *paired bid* has $v_j^i > 0$ and $v_{j'}^i > 0$ for some $j \neq j'$. A *single bid* has $v_j^i > 0$ for some j and $v_{j'}^i = 0$ for all $j' \neq j$.

A bid is *non-generalised* if $\kappa_j^i = k_i$ for all j, or *generalised* if $\kappa_j^i < k_i$ for some j.

A bid is *symmetric* if $a_j^i = a_{j'}^i$ for all j and j', or *asymmetric* otherwise.

An *additional constraint* with label m consists of constants $C_m, C'_m \geq 0$ and coefficients $c_{m,j}^i \geq 0$ for each bid i and good j. These determine a constraint $\sum_{i,j} c_{m,j}^i x_j^i \leq C_m R + C'_m$, where $R = \text{size}(\sigma)$ is the total size of the auction supply.[2]

[2]Thus these additional constraints allow the incorporation of constraints of both the form discussed in the "Additional Constraints" subsection, and the "Constraints on how much individual bidders can win" subsection of the paper "Notes on solving the Standard Version using Linear Programming" on http://pma.nuff.ox.ac.uk.

Note that if $c_{m,j}^i$ depend on the (goods) j, the constraints may create complements, so our program maximises welfare but there may not be a unique lowest competitive equilibrium price. Even if the $c_{m,j}^i$ do not depend on j, but $C_m \neq 0$, prices may not be monotonic in the "size" of the auction, so that the TQSS may be "satisfied" at multiple price vectors (see discussion below).

In fact, we have currently implemented only the case in which $c_{m,j}^i = 1$ for all i, j, m (and only in the command-line version of the software). Specifically, we consider constraints which limit every bidder's total quantity won across all bids *either* to a given quantity C' (i.e. $C_m = 0, C'_m = C'$), *or* to a given fraction C of the auction "size" (i.e. $C_m = C, C'_m = 0$).

1.3 Auction

An *auction* consists of:

- a supply σ;
- a set of bids;
- a set of additional constraints on the bids;
- a scale factor $\rho \in \mathbb{Z}$ giving the number of decimal places of precision required in quantities (prices are always given to the nearest integer);
- a preference permutation $(p_1, \ldots, p_N)$ of $(1, \ldots, N)$ such that $p_j = 1$ for the good that the auctioneer most prefers to allocate, and $p_j = N$ for the good that the auctioneer prefers to allocate last.

To ensure that rounding does not lose information when reporting quantities, the supply and bids should satisfy the invariant that all s_j^q, k_i/a_j^i and κ_j^i/a_j^i are all multiples of $10^{-\rho}$.

2 Solving an auction

Solving an auction requires:

- determining an auction price for each good in the supply (and possibly also the total quantity to be supplied, if a TQSS is in use);
- allocating quantities of goods to successful bids.

2.1 Determining auction prices (single iteration)

Consider the linear programme with variables

- x_j^i (allocation of good j to bid i)
- y_j^q (allocation on q^{th} step of goods j and above)

and objective function

$$\sum_{i,j} \hat{v}_j^i x_j^i - \sum_{j,q} \mu_j^q y_j^q$$

subject to the constraints

$$\begin{array}{rcll}
\sum_j a_j^i x_j^i & \leq & k_i & \forall i \quad \text{(bid constraint)} \\
0 \leq a_j^i x_j^i & \leq & \kappa_j^i & \forall i, j \quad \text{(generalised bid constraint)} \\
0 \leq y_j^q & \leq & \hat{s}_j^q & \forall q, j \quad \text{(step length constraint)} \\
\sum_i x_j^i & \leq & \sum_q y_j^q - \sum_{j' \in \text{Covers}(j)} \sum_q y_{j'}^q & \forall j \quad \text{(good } j \text{ consistency constraint)} \\
\sum_{i,j} c_{m,j}^i x_j^i & \leq & C_m R + C'_m & \forall m \quad \text{(additional constraint } m\text{)}
\end{array}$$

where

$$\hat{v}_j^i = v_j^i + \epsilon^{p_j+1} \qquad \epsilon = \tfrac{1}{2} \qquad \text{(prices tweak)}$$

$$\hat{s}_j^q = \begin{cases} s_j^1 + T_j\eta, & \text{if } q = 1 \\ s_j^q, & \text{otherwise} \end{cases} \qquad \eta = \tfrac{1}{4 \cdot 10^{\rho} \cdot N} \qquad \text{(quantities tweak)}$$

$$T_j = 1 + \textstyle\sum_{j' \in \text{Covers}(j)} T_{j'}$$

$$R = \text{size}(\sigma)$$

To reduce the incidence of ambiguous prices, in addition to the bids provided by the bidders, we add for each good j an "extra bid" with label $j*$, given by (for all goods j'):

$$k_{j*} = \frac{\eta}{2}$$

$$\kappa_{j'}^{j*} = \frac{\eta}{2}$$

$$a_{j'}^{j*} = 1$$

$$v_{j'}^{j*} = \begin{cases} V, \text{if } j = j' \\ 0, \text{otherwise} \end{cases}$$

where V is larger than any other bid price.

Let z_j denote the shadow value of the good j consistency constraint. This represents the price of good j assuming uniform highest-loser pricing.

Solving this linear program yields values for the prices $\mathbf{z} = (z_1, \ldots, z_N)$, and additionally yields an efficient (but not necessarily fair) allocation x_j^i of goods to bids.

To eliminate the effect of tweaks, the calculated prices must be rounded to the nearest natural number, and the calculated quantities must be rounded to ρ decimal places (i.e. to the nearest multiple of $10^{-\rho}$).

2.2 Determining auction prices (with TQSS)

A *TQSS* $\tau \in \mathbb{R}^N \to \mathbb{R}$ is a function from prices of all the goods to the total quantity that should be made available at those prices. It must be (weakly) increasing (i.e. increasing the price of any good must not decrease τ). In general, τ will be a (weakly) increasing function of the price for a particular good, or the total price of all the goods.

Fix $\lambda \in [0, 1]$. Define a supply $\sigma \cdot R$ (σ *scaled to size* R) by multiplying each s_j^q by

- $\frac{R}{R_0}$, if j is a base good
- $\frac{\lambda R_0 + (1-\lambda)R}{R_0}$, otherwise

where $R_0 = \text{size}(\sigma)$. Observe that $\text{size}(\sigma \cdot R) = R$. Note that if $\lambda = 0$ then supply curves for all goods are scaled by $\frac{R}{R_0}$, while if $\lambda = 1$ then supply curves for base goods are scaled but those for non-base goods are left unchanged.

To ensure that the tweaks to determine unique prices are not affected by scaling the supply, the step widths must be limited to ρ decimal places (i.e. must be multiples of $10^{-\rho}$). We round each partial sum of step widths upwards to ρ decimal places. That is, we round the scaled s_j^1 to $\lceil s_j^1 \rceil^\rho$ and the scaled s_j^q to $\lceil \sum_{r=1}^{q} s_j^r \rceil^\rho - \lceil \sum_{r=1}^{q-1} s_j^r \rceil^\rho$, where $\lceil \cdot \rceil^\rho$ denotes rounding upwards to ρ decimal places. This may amplify rounding errors resulting from floating-point imprecision, of course.

The TQSS τ is *satisfied* by the auction with the supply scaled to size R if the prices $\mathbf{z}(R)$ arising from solving the LP are such that $\tau(\mathbf{z}(R)) = R$, i.e. the available supply meets demand.

Our objective is to find R that satisfies the TQSS. In the basic problem, the prices always (weakly) decrease with increasing supply.[3] Suppose we have an auction with supply of size $R_{\min}$, and we also have $R_{\max} > R_{\min}$ such that $\tau(\mathbf{z}(R_{\min})) \geq R_{\min}$ and $\tau(\mathbf{z}(R_{\max})) \leq R_{\max}$. Thus we can perform binary search for a root of $\tau(\mathbf{z}(R)) - R$. If τ is strictly increasing, there will be a unique solution. If τ is weakly increasing, there will be at least one solution, which may lie in an interval (and there will be at most a single interval of solutions).

Thus solving the TQSS yields a final supply size R (accurate to ρ decimal places) in addition to the prices and allocation. For the sequel, all uses of the supply (e.g. when re-running the LP for rationing purposes) are implicitly assumed to have scaled it to size R.

2.2.1 Alternative version of TQSS[4]

As an alternative way of solving the TQSS, rather than varying the supply curves, we can add a constraint

$$\sum_{j \text{ base good}} \sum_{q} y_j^q \leq R + 2N\eta$$

then vary R and search for a solution as before.

With this approach, the maximum value for R that will affect the resulting prices is $\text{size}(\sigma)$.

[3] However, if $C_m \neq 0$, (i.e., there are additional constraints that are a function of R) prices may not always (weakly) decrease with increasing supply, so the TQSS may be "satisfied" at multiple price vectors. However, the program will ensure that the TQSS starts at 0, and covers a large enough range that binary search always yields a solution. (It would of course be possible to use a slower search method to look for, e.g., the smallest R that is (sufficiently close to) a solution.)

[4] See "Program 1* [Alternative version of TQSS]" in section 5 of the paper "Notes on solving the Standard Version using Linear Programming" on http://pma.nuff.ox.ac.uk.

2.3 Allocating goods to bids (rationing)

Once the auction supply size R, prices of each good z_j and total quantity sold of each good R_j have been determined, we must allocate quantities of goods to bids, such that each bid receives its preferences at the given prices. If the bid value for a good is below the auction price, the bid does not receive any of that good. If the bid value is greater than or equal to the auction price, the bid may receive a quantity of the good, but if the supply of the good is limited and the bid is marginal then the bid may receive a smaller quantity than it requested.

Without rationing, the amount of good j allocated to bid i is simply the value of x_j^i arising from solving the LP, rounded to ρ decimal places. This allocation is guaranteed to satisfy the constraints, up to rounding error, but it resolves ties arbitrarily.

2.3.1 Definitions

For each bid i and good j, let

$$w_j^i = \frac{v_j^i - z_j}{a_j^i}$$

be the surplus achieved by allocating that good to that bid.

A bid is *successful* if $\max_j w_j^i \geq 0$, or *unsuccessful* otherwise.

The *relevant bids* of an additional constraint $\sum_{i,j} c_{m,j}^i x_j^i \leq C_m R + C'_m$ are the bids i for which $c_{m,j}^i \neq 0$ for some j. A bid is *constrained* if it is generalised or it is a relevant bid for an additional constraint; otherwise, it is *unconstrained*. Two constrained bids i and i' are *related* if $i = i'$ or if there is an additional constraint for which both i and i' are relevant bids.

A successful unconstrained bid i is *multiply-marginal (on good j)* if $j \in J$ and $|J| \geq 2$, where $J = \operatorname{argmax}_{j'} w_{j'}^i$. That is, the bid is marginal between receiving any of the goods in J, each of which yields equal maximal surplus. (If the surplus is zero, the bid is also marginal between receiving one of the goods and receiving no good at all.)

A successful constrained bid i is *multiply-marginal (on good j)* if $w_j^i = 0$, or if $w_j^i = w_{j'}^{i'}$ for some related bid i' and some good j'. (Note that this definition might be seen as a misnomer: a "multiply-marginal" generalised bid may actually be marginal only on one good, if it is succcessful on another good,[5] but it does not hurt to include it.)

A successful bid is *singly-marginal (on good j)* if is not multiply-marginal and $w_j^i = \max_{j'} w_{j'}^i = 0$. That is, the bid is marginal on receiving or not receiving precisely one good yielding zero surplus, and does not receive any other goods.

A successful bid is *marginal (on good j)* if it is singly or multiply-marginal (on good j).

[5] The reason is that, if additional constraints apply, the bid may be marginal between receiving and not receiving a good yielding zero surplus (and not marginal between receiving either of a pair of goods of equal surplus).

2.3.2 Reformulating the LP

Observe that unsuccessful bids always receive zero allocation, and bids that are successful and not marginal always receive a full allocation of their desired good. This is automatic in the LP. However, for a marginal bid there may not be enough goods to fulfill the bid entirely, so it may receive less than its desired allocation (including no units at all). The LP will allocate goods to marginal bids arbitrarily, which is undesirable.

Thus we reformulate the problem to produce a fairer allocation. We proceed by adjusting the multiply-marginal bids, solving a new LP to find the allocations, then fairly redistributing between sets of singly-marginal or equivalent bids. This approach may slightly favour paired bids.[6]

Replace each multiply-marginal bid i with a family of bids (i, l) approximating a linear demand that decreases from k_i units at value v_j^i to 0 units at value $v_j^i + j\delta$, where $\delta = \frac{1}{N+1}$. Solve the auction LP with the total quantity of each good supplied now fixed at $R_j = \sum_i x_j^i$, and without the tweaks to determine unique prices/quantities, to produce allocations $x_j^{(i,l)}$.

More precisely, let d_i be the number of steps with which to approximate bid i, and let the family of bids replacing bid i be $(i, 0), \ldots, (i, d_i - 1)$ where

$$\begin{aligned}
k_{(i,l)} &= k_i/d_i \\
\kappa_j^{(i,l)} &= \kappa_j^i/d_i \\
v_j^{(i,l)} &= \begin{cases} v_j^i + (l/d_i)j\delta, & \text{if bid } i \text{ is marginal on good } j \\ v_j^i, & \text{otherwise} \end{cases} \\
a_j^{(i,l)} &= a_j^i
\end{aligned}$$

Fix D and let $d_i = D$ if bid i is multiply-marginal, and $d_i = 1$ otherwise.[7]

Correspondingly, we need to replace each additional constraint

$$\sum_{i,j} c_{m,j}^i x_j^i \leq C_m R + C'_m$$

with

$$\sum_{i,l,j} c_{m,j}^{(i,l)} x_j^{(i,l)} \leq C_m R + C'_m$$

where

$$c_{m,j}^{(i,l)} = c_{m,j}^i/d_i$$

[6] It is also possible to adjust all marginal bids, not just those that are multiply-marginal. This produces a more complex LP, but avoids the need to redistribute between singly-marginal bids, and does not favour paired bids in the same way.

[7] If bids are divided up into too many steps, the program might encounter arithmetic precision issues. There is a trade-off between having enough steps to minimize the impact of arbitrary choices, but not so many that it becomes impossible to distinguish the original prices from the slightly increased prices. The (command-line version of the) software permits the user to choose the step count, D, explicitly if wished.

We then solve the linear programme with variables

$$x_j^{(i,l)} \quad \text{(allocation of good } j \text{ to bid } i \text{ on step } l\text{)}$$

and objective function

$$\sum_{i,j,l} v_j^{(i,l)} x_j^{(i,l)}$$

subject to the constraints

$$\begin{array}{rcll}
\sum_j a_j^i x_j^{(i,l)} & \leq & k_{(i,l)} & \forall i,l \quad \text{(bid constraint)} \\
0 \leq a_j^i x_j^{(i,l)} & \leq & \kappa_j^{(i,l)} & \forall i,j,l \quad \text{(generalised bid constraint)} \\
\sum_{i,l} x_j^{(i,l)} & \leq & R_j & \forall j \quad \text{(resource constraint for good } j\text{)} \\
\sum_{i,j,l} c_{m,j}^{(i,l)} x_j^{(i,l)} & \leq & C_m R + C'_m & \forall l \quad \text{(additional constraint } m\text{)}
\end{array}$$

We now need to specify how to calculate fair allocations for the original bids.

2.3.3 Fairly sharing between singly-marginal bids

Let the initial interim allocation of good j to bid i be

$$x'^{\,i}_{\ j} = \sum_l x_j^{(i,l)}$$

This should now be fair between multiply-marginal bids, but singly-marginal bids may still have arbitrarily received or not received units of their single preferred good, and equivalent bids may not have received equal allocations.

Thus we redistribute the units allocated to singly-marginal bids fairly between them in proportion to the quantities they requested. More precisely, the reallocation is:

$$x''^{\,i}_{\ j} = \begin{cases} \kappa_j^i \frac{\sum_{i_m} x'^{\,i_m}_{\ j}}{\sum_{i_m} \kappa_j^{i_m}} & \text{if bid } i \text{ is singly-marginal on good } j \\ x'^{\,i}_{\ j} & \text{otherwise} \end{cases}$$

(where i_m ranges over the bids that are singly-marginal on good j).

Observe that bids subject to additional constraints are not singly-marginal, and hence this redistribution cannot violate the additional constraints.

2.3.4 Fairly sharing between equivalent bids

Finally, we wish to ensure that equivalent bids get equal shares: if two bids each offer the same price for the same quantity, we should not give a full allocation to one and ration the other. In the absence of additional constraints, equivalence simply requires that bids are identical apart from their labels. However, in the presence of additional constraints, it is nontrivial to specify a suitable equivalence relation (we would not want to consider bids equivalent if they were

subject to meaningfully different constraints). Thus we simplify slightly and treat all bids subject to additional constraints as distinct.

A bid that is relevant for an additional constraint is *equivalent* (only) to itself. Two bids, neither of which is relevant for an additional constraint, are *equivalent* if they are identical apart from their labels.

To ensure that equivalent bids get equal shares: for each non-singleton equivalence class, take the total interim allocation to the whole class of bids and share it evenly between them (rounding every allocation down to a multiple of $10^{-\rho}$). That is,

$$x'''^{\,i}_{\,j} = \left\lfloor \frac{\sum_{i_n} x''^{\,i_n}_{\,j}}{n} \right\rfloor^{\rho}$$

(where $i_1, \ldots, i_n$ are the bids that are equivalent to bid i). We round down here to avoid producing results that allocate too much, e.g. if sharing 5 units between 3 bids demanding 2 units each, rounded to the nearest integer, we should report 1 unit per bid rather than 2 units per bid.

This will not violate any additional constraints, because the amount allocated to a bid after sharing will never be more than the amount previously allocated to an equivalent bid (which is subject to corresponding constraints).

2.3.5 Choice of step count to limit unfairness

The maximum unfair share (i.e. the amount by which one multiply-marginal bid may be arbitrarily favoured over another) is no more than $u = \max_i k_i/D$, where i ranges over multiply-marginal bids. This is because unfairness is limited by the width of the steps into which we break down the bids: the worst that can happen is that a step is fully used on one bid but not on others.

Choosing $D = 1 + \max_i k_i \cdot 2 \cdot 10^{\rho}$ ensures that $u < 1/(2 \cdot 10^{\rho})$ and hence rounding to the nearest multiple of $10^{-\rho}$ will limit the arbitrary unfairness to at most a difference of $10^{-\rho}$. For example, if $\rho = 2$ and we are sharing 1 unit between 3 bidders, we might report allocations of 0.33, 0.33 and 0.34, but not 0.3, 0.3 and 0.4.[8]

[8] Of course, if the bids are identical in every respect apart from their labels, then "Fairly sharing between equivalent bids" (previous subsection) will reduce them all to 0.33, 0.33, 0.33.