# Net pool versus gross pool market model formulations

John F. Raffensperger

1 February 2009, updated 3 Aug 2009.

Notes for my students. This information does not appear to be documented elsewhere.

## Market models can be formulated as net or gross pool

A gross pool formulation is one in which the model clears all rights, not just the net buying and selling. Users provide bids for the total quantity they wish to have, regardless of initial rights.

In contrast, a net pool formulation clears only the buying and selling. Users provide bids only for the changes from their existing initial rights.

One sided auctions would be formulated as gross pool, because the market manager is serving as either the sole buyer or the sole seller. Two sided auctions, however, can be formulated as either net or gross pool. So we are discussing the two-sided auctions here.

## The net pool formulation.

**Algebraically, the net pool formulation can be given as:**

Maximize $\sum_{\text{users } i}\sum_{\text{buy bids } b} Pbuy_{ib}\, buy_{ib} - \sum_{\text{users } i}\sum_{\text{sell bids } b} Psell_{ib}\, sell_{ib}$,

$q_i - \sum_{\text{buy bids } b} buy_{ib} + \sum_{\text{sell bids } b} sell_{ib} = InitialRight_i$ for each user $i$,

$buy_{ib} \leq Buylimit_{ib}$, $sell_{ib} \leq Selllimit_{ib}$ for each user $i$, and tranches $b$,

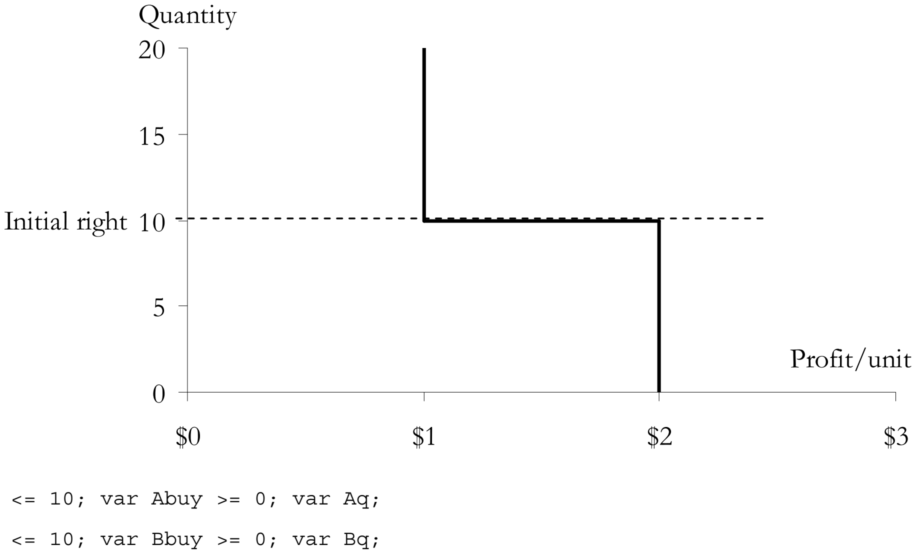
$Aq \leq d$, some arbitrary constraints in generic matrix form.

Note that if users refuse to sell, then $sell_{ib}$ variables are all 0, and $q_i$ could be too big, and the model could be infeasible. So some market rules must be established to ensure that the market manager can operate the auction realistically.

We can interpret the objective coefficients as the marginal profit to the user for a change from the user's initial right.

**Example**

Consider the following net pool model. Each user has a value of $2/unit for 0 to 10 units, and a value of $1/unit for more than 10 units. Each user has initial rights to 10 units.



```
var Asell >= 0 <= 10; var Abuy >= 0; var Aq;
var Bsell >= 0 <= 10; var Bbuy >= 0; var Bq;
```

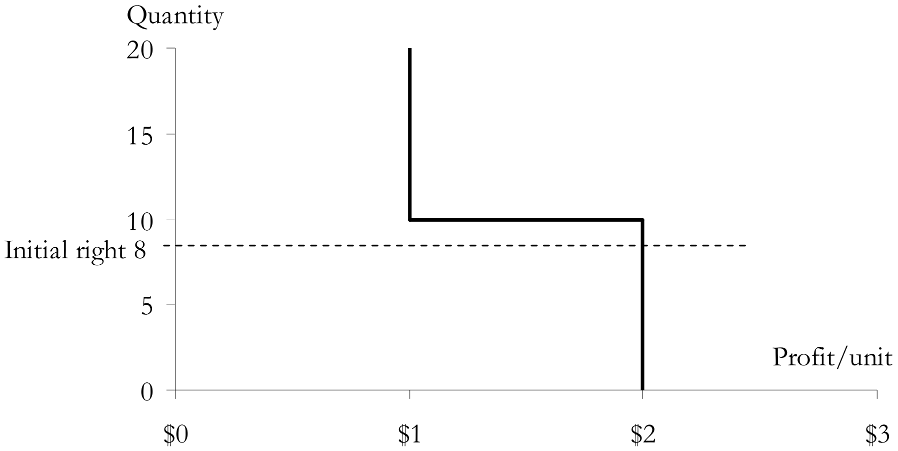
```
maximize Welfare: Abuy - 2*Asell + Bbuy - 2*Bsell;
subject to Atotal: Aq - Abuy + Asell = 10;
subject to Btotal: Bq - Bbuy + Bsell = 10;
subject to Environment: 0.1*Aq + Bq <= 9;

solve;
printf "\tBuy\tSell\tDisch\tPrice\n";
printf "A\t%g\t%g\t%g\t%g\n", Abuy, Asell, Aq, Atotal.dual;
printf "B\t%g\t%g\t%g\t%g\n", Bbuy, Bsell, Bq, Btotal.dual;
printf "Net to govt = %g.\n", -Atotal.dual*(Abuy - Asell) + -Btotal.dual*(Bbuy - Bsell);
```

```
        Buy     Sell    Disch   Price
A       80      0       90      -1
B       0       10      0       -10
Net to govt = -20.
```

Result: objective value 60. This is the users' welfare, as user A breaks even on all units, and user B gains 10*($10 – $2) = $80, with –$20 from government.

Now suppose we want to see what happens if the initial rights change from 10 to 8. For example, because we see that government had to pay users, perhaps we reduce α from 100% to 80%. We have to change the whole model, because the users still have a value of $2/unit for 8 to 10 units. This is annoying!



It is not enough to change the right hand sides of Atotal and Btotal, because the upper bounds on Asell and Bsell are still 10. So maybe we could also change the upper bounds of Asell and Bsell to 8? No, again, that is not enough. Units 8 to 10 have flipped from *sell* quantities to *buy* quantities. The objective would give a value of only $1/unit , when they are actually worth $2/unit to each user. Therefore, we have to add two more *buy* variables, worth $2/unit. This is annoying, and difficult to code correctly.

```
var Asell >= 0 <= 8; var Abuy1 >= 0 <= 2; var Abuy2 >= 0; var Aq;
var Bsell >= 0 <= 8; var Bbuy1 >= 0 <= 2; var Bbuy2 >= 0; var Bq;
maximize Welfare: 2*Abuy1 + Abuy2 - 2*Asell + 2*Bbuy1 + Bbuy2 - 2*Bsell;

subject to Atotal: Aq - Abuy1 - Abuy2 + Asell = 8;
subject to Btotal: Bq - Bbuy1 - Bbuy2 + Bsell = 8;
subject to Environment: 0.1*Aq + Bq <= 9;

solve;
printf "\tBuy\tSell\tDisch\tPrice\n";
printf "A\t%g\t%g\t%g\t%g\n", Abuy1 + Abuy2, Asell, Aq, Atotal.dual;
printf "B\t%g\t%g\t%g\t%g\n", Bbuy1 + Bbuy2, Bsell, Bq, Btotal.dual;
printf "Net to govt = %g.\n", -Atotal.dual*(Abuy1 + Abuy2 - Asell) + -Btotal.dual*(Bbuy1 + Bbuy2 -
Bsell);
        Buy     Sell    Disch   Price
A       82      0       90      -1
B       0       8       0       -10
Net to govt = 2.
```

Result: objective value 68. Again, this is users' welfare: user A gains 2*($2 – $1) on the first two units purchased and breaks even on the last 80 units purchased, while user B gains 8*($10 – $2), with $2 to government.

## Gross pool formulation

**Algebraically, the gross pool formulation can be given as:**

Maximize $\sum_{\text{users } i}\sum_{\text{bids } b} P_{ib}\ bid_{ib}$,

$q_i = \sum_{\text{bids } b} bid_{ib}$ for each user $i$,

$bid_{ib} \leq Bidlimit_{ib}$ for each user $i$ and bid $b$,

$Aq \leq d$, some arbitrary constraints in generic matrix form.

Now then, *after* solution of the model, we can calculate the net sales.

$buy_i = \max(0, q_i - InitialRight_i)$ for each user $i$.

$sell_i = \max(0, InitialRight_i - q_i)$ for each user $i$.

Note that the gross pool formulation is always feasible (for matrix $A \geq 0$, as it is for ground water). To find a feasible solution, we can just set the $bid_{ib}$ variables to zero. Users who do not want to sell would want to set their relevant $P_{ib}$ coefficients to some unrealistically large value. So instead of being mathematically infeasible, the objective value is finite, but probably too big for the market manager. So some market rules must be established, as with the net pool formulation, to ensure that the market manager can operate the auction realistically.

We can interpret the objective coefficients as the marginal profit to the user for each tranche of water, independent of the user's initial right.

By the Coase Theorem, the net and gross pool models must give identical results. But some people don't agree, e.g., http://www.erisk.net/erisk6/article/377487/england_wales_market_shift_to_net_pool_design_over_concerns_about.

**Example**

Now consider the analogous following gross pool market.

```
var Abid1 >= 0 <= 10; var Abid2 >= 0; var Aq; # Aq must be free.
var Bbid1 >= 0 <= 10; var Bbid2 >= 0; var Bq; # Bq must be free.

maximize Welfare: 2*Abid1 + Abid2 + 2*Bbid1 + Bbid2;
subject to Atotal: Aq = Abid1 + Abid2;
subject to Btotal: Bq = Bbid1 + Bbid2;
subject to Environment: 0.1*Aq + Bq <= 9;

solve;
param Abuy; let Abuy := max (0, Abid1 + Abid2 - 10);
param Bbuy; let Bbuy := max (0, Bbid1 + Bbid2 - 10);
param Asell; let Asell := max (0, 10 - Abid1 - Abid2);
param Bsell; let Bsell := max (0, 10 - Bbid1 - Bbid2);

printf "\tBuy\tSell\tDisch\tPrice\n";
printf "A\t%g\t%g\t%g\t%g\n", Abuy, Asell, Aq, Atotal.dual;
printf "B\t%g\t%g\t%g\t%g\n", Bbuy, Bsell, Bq, Btotal.dual;
printf "Net to govt = %g.\n", -Atotal.dual*(Abuy - Asell) + -Btotal.dual*(Bbuy - Bsell);
        Buy     Sell    Disch   Price
A       80      0       90      -1
B       0       10      0       -10
Net to govt = -20.
```

Result: objective value 100. This is users' total profit from water, ignoring their initial right. User A gains $2/unit on the first 10 units and $1/unit on the remaining 80. User B gains nothing, as she used no water. Of course, money changed hands during settlement, but the linear program was ignorant of the initial rights.

Note that we calculate the buys and sells *after* solution.

Now suppose we want to see what happens if the initial rights change from 10 to 8. We do not have to touch the model at all! We simply calculate the difference at the end. Using the same gross pool model as above, we calculate the buys and sells, again after solution.

```
var Abid1 >= 0 <= 10; var Abid2 >= 0; var Aq; # Aq must be free.
var Bbid1 >= 0 <= 10; var Bbid2 >= 0; var Bq; # Bq must be free.
maximize Welfare: 2*Abid1 + Abid2 + 2*Bbid1 + Bbid2;

subject to Atotal: Aq = Abid1 + Abid2;
subject to Btotal: Bq = Bbid1 + Bbid2;
subject to Environment: 0.1*Aq + Bq <= 9;
```

Same as above model

```
solve;
param Abuy; let Abuy := max (0, Abid1 + Abid2 - 8);
param Bbuy; let Bbuy := max (0, Bbid1 + Bbid2 - 8);
param Asell; let Asell := max (0, 8 - Abid1 - Abid2);
param Bsell; let Bsell := max (0, 8 - Bbid1 - Bbid2);

printf "\tBuy\tSell\tDisch\tPrice\n";
printf "A\t%g\t%g\t%g\t%g\n", Abuy, Asell, Aq, Atotal.dual;
printf "B\t%g\t%g\t%g\t%g\n", Bbuy, Bsell, Bq, Btotal.dual;
printf "Net to govt = %g.\n", -Atotal.dual*(Abuy - Asell) + -Btotal.dual*(Bbuy - Bsell);
        Buy     Sell    Disch   Price
A       82      0       90      -1
B       0       8       0       -10
Net to govt = 2.
```

Result: objective value $100. As before, this is users' total profit from water, ignoring their initial right. User A gains $2/unit on the first 10 units and $1/unit on the remaining 80. User B gains nothing, as she used no water.

## Conclusion

So there is no difference between the gross and net pool formulations theoretically. However, the gross pool model is probably easier to manage from:

- an operational point of view, as the model is always feasible, so the auction manager never has to deal with a broken model,
- database point of view, because the programmer can ignore initial rights during formulation,
- and experimentation, as we have no need to change the model to change the initial right.

Especially when we start adjusting the $\alpha$ parameter, which controls initial rights, we will be glad to use the gross pool formulation! Every change in $\alpha$ would require modification of the bid variables, objective coefficients, and bounds.

The gross pool formulation is always feasible mathematically, and we do not have to worry about buys, sells, or initial rights within the model, because those are easily handled after the fact as part of the financial settlement process.

## Acknowledgements

Thanks to Grant Read for explaining much of this to me. Thanks to Antonio Pinto for finding mistakes. The examples and any errors are my own.