Optimal item price

- We do not know, given an instance, what is the optimal item price p^{OPT}.

- However we can identify a set of prices P of polynomial size that contains p^{OPT}. Let us see how to construct such set P.

- We are going to define a set of prices P_i for each buyer $i=1,\ldots,n$.

- Informally, P_i is the set of prices for which if we increase any of them by a very small quantity then the set of bundles that maximizes the utility of buyer i changes.

Optimal item price (2)

- An easy way to get P_i is to compute, for any pair $j,j' \in \{0,1,\ldots,m\}$ where $j \neq j'$, the price p such that $v_i(j) - pj = v_i(j') - pj'$ and include p in P_i if $p \geq 0$.
  - For example when $j'=0$, we have all the prices $p \geq 0$ such that $v_i(j) - pj = 0$.
- Notice that the number of different prices belonging to P_i is $O(m^2)$
- We define $P = \bigcup_{i \in \{1,\ldots,n\}} P_i$
- Notice that the number of different prices belonging to P is $O(nm^2)$ that is polynomial in the input size.

Lemma B: *Given an instance of the pricing problem, the optimal item price* $p^{OPT} \in P$.

Proof (Idea):

Given an optimal solution, it is possible to show by contradiction that if $p^{OPT} \notin P$, then we can sell the same number of items as the optimal allocation (indeed the same allocation of the optimal outcome) but at an higher item price. This would give an higher revenue contradicting that it was an optimal solution. □

Approximation algorithms.

- We are going to design a polynomial time algorithm that finds approximated solutions for our problem.

We now roughly define what is an approximation algorithm for our problem:

- Let us suppose we have designed a polynomial time* algorithm that finds a feasible and envy-free solution. Let *SOL* be the value of the solution returned by our algorithm and let *OPT* be the value of an optimal solution to the problem.
- Recall that we are considering a maximization problem!

- We say that our algorithm is an *r-approximation* algorithm (where $r \leq 1$) for the problem, if the algorithm produces for any input a solution such that:

$$\frac{SOL}{OPT} \geq r$$

* Notice that the definition of approximation algorithms is not restricted to polynomial time algorithms. However we are only interested in polynomial time algorithms.

A polynomial time approximation algorithm

- In the next slide we are going to see a polynomial time approximation algorithm.

- In the following algorithm when we say ‘buyer *i* demands *j* item’ we mean that, given an item price *p,* buyer *i* demands *j* items if the bundle composed by *j* items is the one that gives him the maximum utility.

- The following algorithm considers all the prices belonging to *P,* and, for each $p \in P$, computes a feasible and envy-free outcome which is a logarithmic approximation of the optimal outcome for that *p.* It finally returns the feasible and envy-free outcome that achieves the highest revenue.

A polynomial time approximation algorithm

ALGORITHM 1: A logarithmic approximation algorithm for the pricing problem.

for $p \in P$ do

  For any $j \in \{1, \ldots, m\}$, let n_j be the number of buyers such that the bundle of j items is the largest one they have non-negative utilities (i.e., greater or equal than zero utilities) for at price p;

  Sell bundles with sizes at least z where $z = \text{argmax}_j \, j \sum_{k=j}^{m} n_k$;

  Let $\bar{n}_j$ be the number of buyers who demand j items at price p (here we suppose that a buyer prefers a non-empty bundle giving her utility zero with respect to buying no item) if only bundles of at least z items are sold;

  if $\sum_{j \geq z}^{m} j\bar{n}_j \leq m$ then

    Assign $j \geq z$ items to the buyers who demand j items and charge each of them pj;

  end

  else

    Apply Lemma A to obtain a feasible outcome;

  end

end

Output the outcome with the maximum revenue.