Combinatorial Auctions - Fall 2020
Assignment #6: Combinatorial Auctions

Professor Sven Seuken
Department of Informatics, University of Zurich
Out Wednesday, November 18, 2020
Due **12:15** sharp: **Wednesday, November 25, 2020**
For submission format, check description below.

[**Total: 100 Points**] This is a single-person assignment. Points will be awarded for clarity, correctness and completeness of the answers. Reasoning must be provided with every answer, i.e., please show your work. You get most of the credit for showing the way in which you arrived at the solution, not for the final answer. You are free to discuss the assignment with other students. However, you are not allowed to share (even partial) answers and source code with each other, and **copying will be penalized**.

Your submission should be made via Moodle as a single .zip file containing (i) your written answers to all questions in one PDF file, and (ii) your code (i.e., all .java files). The submissions will be closed on Wednesday, November 25, 2020 at 12:15.

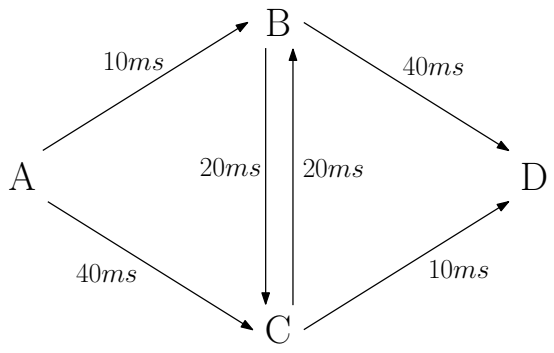
1 Notes and Setup

In this assignment, you will implement a combinatorial auction using Java/JOpt, which you have installed for previous exercise sheets. For all programming exercises, we provide you with a code skeleton. To set everything up, you need to first download the source code from our course homepage and import it into Eclipse.

1. Download the materials for this assignment from Moodle. Included is a code skeleton for problems 2 and 3.
2. In Eclipse, go to "File" $\rightarrow$ "Import" and select "Existing Maven Projects" from the list of import wizards. Then choose the code skeleton folder as the root directory.
3. The code skeleton comes without a main method, but there is a small test suite you can use to test your code. Please note that your methods also have to work for other test samples (not only the test cases we provide).
4. You can add new classes, methods, etc. However, it must be possible to execute your code by calling the methods we provided with the same arguments. (We will use automated testing to check the correctness of your solution.)
5. Have fun!

2 Problem Set

1. **[25 Points]** Combinatorial Auctions: Bidding Languages



Two agents can bid for computer network links. Each of the agents wishes to send a message from A to D. Each link can carry one message in the direction specified by the corresponding arrow, taking a certain number of milliseconds to be traversed. Agent 1 has a value of $100/t_1$ for his message to arrive at D, where t_1 is the total time his message takes to arrive. Similarly, agent 2 has a value of $60/t_2$ for his message to arrive at D.

Every agent needs to send a single message and thus can use only a single path. Agents can bid on the exclusive right to use different links, that is the goods up for auction are the links `AB, AC, BC, CB, BD` and `CD`.

(a) **[10 Points]** For each agent, formulate an XOR bid that expresses their true valuation of all possible bundles. Do the same for the OR* bidding language.

(b) **[8 Points]** Prove that the true valuations of these agents cannot be expressed in the OR bidding language.

(c) **[7 Points]** Find the efficient allocation and the payments each bidder would have to make under the VCG mechanism.

2. **[40 Points]** Winner Determination

In this task you need to implement an algorithm to solve the winner determination problem in a combinatorial auction (see lecture notes, Definition 11.27), assuming that bids are submitted using the XOR-bidding language.

(a) **[20 Points]** Formulate the auctioneer's winner determination problem as an integer program, i.e., formalize all variables, constraints, and the objective. Your formulation should cover all possible auctions where n bidders are bidding for m different goods.

(b) **[20 Points]**
Use JOpt to implement the integer program from (a), using the template provided in `src/main/java/.../winnerdetermination/XORWinnerDetermination.java`.
Verify your implementation with the tests that can be found in the package `src/test/java/.../winnerdetermination`.

3. **[35 Points]** VCG Mechanism

   In this task you need to implement the VCG mechanism for combinatorial auctions. Using your code for the winner determination problem from the previous task, you need to implement the VCG payment rule.

   (a) **[15 Points]** Implement the VCG mechanism. A code skeleton is provided in the class `src/main/java/.../vcg/VCGAuction.java`. You only need to implement the payment rule. For the allocation, re-use the code from task 2.

   Verify your implementation with the tests that can be found in the package `src/test/java/.../vcg`.

   (b) **[12 Points]** Implement the VCG mechanism with reserve prices: suppose there is a per-item reserve price $r_a > 0$ for each item $a \in G$, and the reserve price for any bundle is equal to the sum of the reserve prices of the items in the bundle, i.e., $r_B = \sum_{a \in B} r_a$ for all bundles $B \subseteq G$. For the purpose of this question we assume that the reserve prices of goods are all equal (i.e. $r_a = r$ for all $a \in G$).

   Reserve prices work as follows: all atomic bids that violate the reserve price of the corresponding bundle are eliminated. The payment rule is then adjusted such that winning bidders pay the maximum of their actual VCG payment and the reserve price for the bundle they receive.

   Implement VCG with reserve prices using the template provided in the class `src/main/java/.../vcg/ReservePriceVCGAuction.java`.

   Verify your implementation with the tests that can be found in the package `src/test/java/.../vcg`.

   (c) **[8 Points]** Consider the following variant of implementing reserve prices: There is a dummy bidder that bids the reserve price (as defined above) on each bundle of goods. The dummy bidder is treated like a regular bidder for the purpose of calculating VCG payments, but the bundle of goods he wins, if any, stays with the auctioneer.

   Find a set of goods and bids on those goods such that this variant produces a different auction result than the mechanism from part b).

   (**Hint:** You only need 2 non-dummy bidders and 2 goods)