Homework 6: report

Haukur, Grzegorz

March 28, 2017

1 Problem setup

The problem that we solved was to compute the nucleolus of a given bankruptcy game. The bankruptcy game is specified as a vector of positive numbers $d_1, \ldots, d_n$ representing debt to specific players and a positive number E corresponding to a value which can be divided. The surplus of a given coalition C is defined as $max(0, E - \Sigma_{i \in N \setminus C}\ d_i)$ and the nucleolus is defined as a division of E between the players minimizing the lexicographical ordering of complaints over all coalitions. Complaint of a coalition C is defined as $v(C) - \Sigma_{i \in C} x_i$. This division is an imputation, $\mathbf{x}$, if all complaints are less than 0.

2 Implementation

Our program takes 3 objects as an input. The first two define the TU game as an input a vector of positive real numbers corresponding to debts, $\mathbf{d}$, and a positive number corresponding to divisible goods, E. The length of $\mathbf{d}$ determines the number of players N and all possible coalitions $C \subseteq N$. From these we compute the surplus, $v(C)$, of all $C \subseteq N$. The last input, x_s, is an arbitrary vector of length N summing up to E. From this initialization the complaints of all coalitions are computed. Now our algorithm starts on step 1.

Our algorithm consists of two main steps. Firstly, in each iteration we define a value, C, which will be taken from the imputation of player i and given to player j. We choose the players s.t. player i is not in all coalitions which have complaints smaller or equal to $HC - C$, where HC is the highest complaint and C is the iteration constant. Player j is then chosen, arbitrarily, from one of the coalitions which has the highest complaint. When there is no player which satisfies the condition for player i, we move to the next step.

Secondly, we go through all coalitions which have complaints smaller or equal to $HC - C$ and it is checked whether the lexicographical order of all complaints can be improved by taking C from one player within these coalition and giving it to another distinct player within the same coalition. We do this by going through each of those coalitions and try all distinct pairs of players. In case this adjustment would improve the lexicographical order of complaints, we perform

this action and get back to the beginning of the second step. The algorithm terminates when no further such improvements can be made.

3 Examples

We used the following examples:

3.1 Example from the lecture

$$TU = ([30, 60, 90], 100)$$

$$x_s : [33, 33, 34]$$

Conclusions:

- complaints: [-42.5, -37.5, -32.5, -27.5, -15.0, -15.0]
- imputation: [15.0, 32.5, 52.5]

3.2 An intuitive example

$$TU = ([30, 30, 30], 89)$$

$$x_s : [89, 0, 0]$$

With C is 0.5:
Conclusions:

- Complaints: [-1.0, -0.5, -0.5, -0.5, -0.5, 0.0]
- Imputation: [29.5, 29.5, 30.0]

With C is 0.3333333:
Conclusions:

- Complaints: [-0.66726, -0.66637, -0.66637, -0.33363, -0.33363, -0.33274]
- Imputation: [29.66726, 29.66637, 29.66637]

3.3 Another short example

$$TU = ([10, 20, 30], 36)$$

$$x_s : [10, 10, 16]$$

Conclusions:

- Complaints: [-14.5, -10.5, -9.5, -9.5, -5.0, -5.0]
- Imputation: [5.0, 10.5, 20.5]

3.4 A complex example

This example took about 10 seconds on our machine.

$$TU = ([30, 60, 90, 19, 10, 12, 10, 10, 10, 20], 100)$$

$$x_s : [100, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Conclusions:

- Complaints: List of complaints is long.
- Imputation: [15.0, 19.5, 20.0, 9.5, 5.0, 6.0, 5.0, 5.0, 5.0, 10.0]