CPS 296.1 - Linear and integer programming

Homework 2: Extending the cooperative game theory example from class. Due Nov. 17.

Please read the rules for assignments on the course web page. Contact Vince (conitzer@cs.duke.edu) with any questions. All questions are worth the same number of points.

In this assignment, we again take up the subject of cooperative game theory. We have seen examples of games where the core is empty. Even in these games, we need to divide the value $v(A)$ in some way. Our goal will be to be *as close as possible* to being in the core. Specifically, for a given $\epsilon \geq 0$, we say that the vector $(\pi(1), \ldots, \pi(n))$ is in the *ϵ-core* if every coalition has at most ϵ incentive to deviate—that is, for every $S \subseteq A$, $v(S) - \sum_{i \in S} \pi(i) \leq \epsilon$. There is some smallest ϵ for which the ϵ-core is nonempty; we want to find this smallest ϵ, as well as a payoff vector in this smallest ϵ-core (also called the *least core*). That is, we want to minimize the greatest incentive for a coalition to break off. (In general, if the core is actually nonempty, people sometimes want to make sure that every coalition (other than the grand one) has a cost of at least $-\epsilon$ for deviating, which results in something stronger than the core if ϵ is negative. However, in this assignment you can focus on minimizing the incentive to deviate when the core is empty. That is, you can assume $\epsilon \geq 0$.)

As an example, consider the characteristic function given by:

- $v(\{a, b, c\}) = 9$
- $v(\{a, b\}) = 7$
- $v(\{a, c\}) = 8$
- $v(\{b, c\}) = 9$
- v is 0 everywhere else.

If we set $\pi(a) = 2$, $\pi(b) = 3$, $\pi(c) = 4$, then we have a solution in the 2-core (every coalition of size 2 could do better by 2 by deviating). This is the best possible, that is, the 2-core is the least core. (To see why, if we increase any agent's payment, then the coalition of the two other agents will have more than 2 incentive to break off.)

1. Give a linear program formulation for the problem of identifying a solution in the least core. (You will use this throughout the assignment, so make it clean and elegant...)

2. Give the dual of this linear program formulation.

3. Consider the following tiny example where $A = \{a, b\}$.

- $v(\{a, b\}) = 10$
- $v(\{a\}) = 5$
- $v(\{b\}) = 7$

Solve this example by using the simplex algorithm (by hand) on your linear program formulation, starting from the (presumably infeasible) dictionary where every original variable is set to 0. You can use whatever pivoting rule you like, and your dictionaries do not necessarily need to be feasible (except the last one!). You do not need to write constraints if you know that they will not matter (be binding).

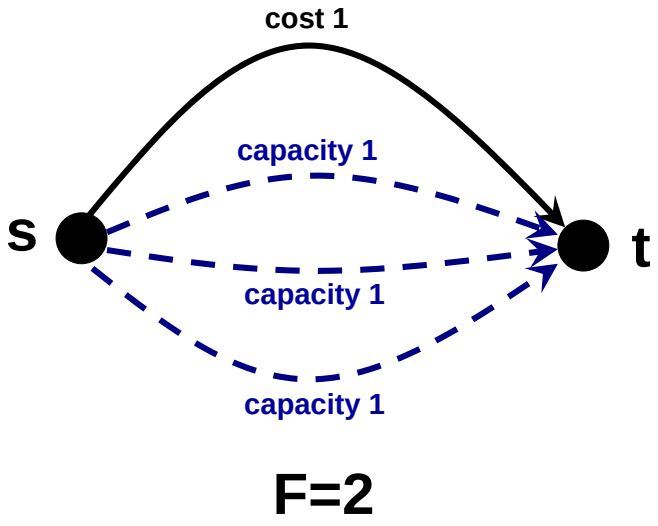

Figure 1: An instance of our problem with an empty core.

Consider again the network-based characteristic function game from class. We can try to extend the techniques that we developed for finding a solution in the core to finding a solution in the least core.

As an example, consider the graph in Figure 1. In this example, we can give each of the agents a payoff of $2/3$. Then, each coalition of size 2 has an incentive of $2 - 2 \cdot 2/3 = 2/3$ to deviate (and the other coalitions have less incentive), so this is a solution in the $2/3$-core, which is the least core for this example.

4. In the context of this network game (in general, that is, not the specific example above), describe an algorithm for generating the most violated constraint in your linear program.

5. For the core, we saw that it was enough to consider only the constraints corresponding to coalitions consisting of a path. Show that this is not enough for the least core. That is, give an example where each path coalition has an incentive of at most ϵ to deviate, but there is another coalition (that does not consist of exactly a single path) that has an incentive of more than ϵ to deviate.

6. For the core, we showed that we did not really need to do constraint generation in the context of the network game: we could use a max flow/min cut approach instead. Generalize this approach to the least core, and prove that your generalized approach is correct.