Homework #1

Deadline: Wednesday, 12 April 2017, 19:00

Submit your solutions for (up to) three of the following four exercises. If you solve all four, we will consult a random number generator to decide which three to look at and grade.

Question 1 (10 marks)

In analogy to the definition of a Condorcet winner, a *Condorcet loser* is an alternative that would lose against every other alternative in a pairwise majority contest.

(a) Give an example that shows that the plurality rule *can* elect a Condorcet loser.

(b) Prove that the Borda rule *never* elects a Condorcet loser.

Remark: It is in fact possible to show that the Borda rule is the *only* positional scoring rule (with a strictly descending scoring vector) that satisfies this property. This is not part of this exercise, but you may still want to think about it.

Question 2 (10 marks)

I ask you to not discuss this particular exercise with each other before the submission deadline, as that would render it pretty much pointless almost immediately.

In the late 1980's, Hervé Moulin published a paper in which he shows that every voting rule that always elects the Condorcet winner when it exists must allow for situations where a voter has an incentive to not vote at all rather than to vote in line with her truthful preferences. Track down this paper and then answer the following questions:

(a) Provide the full bibliographic information for the paper.

(b) Briefly describe how you tracked down the paper.

(c) Give a compact and precise statement of the aforementioned result, using the notation and terminology of the course.

(d) Find one political scientist, one philosopher, and one computer scientist who have cited Moulin's paper in the past 10 years. For each of them, provide full bibliographic information of the contribution in question (including a link), provide evidence that the person in question indeed is a political scientist/philosopher/computer scientist, and describe the reason for the citation in a few words.

Question 3 (10 marks)

Recall that most voting rules are not *resolute*. Therefore, for this exercise, assume that every irresolute voting rule is paired with a *lexicographic tie-breaking rule*. So there always is a single winner. Consider an election with n voters and m alternatives. Suppose alternative x wins. Let us call the *regret* of voter i the number of alternatives that dominate x in her preference order (i.e., this is a number between 0 and $m-1$). We might be interested in the *average regret* a given voting rule F generates for a given profile. Now suppose every voter draws her true preference order from the uniform probability distribution over all $m!$ logically possible orders (this is known as the *impartial culture assumption*). Under this assumption, we can calculate the *expected regret* of voter i under voting rule F. If we average over all voters, we obtain the *average expected regret* of F. Answer the following questions:

(a) Suppose you want to use a voting rule that minimises average regret. Which of the rules discussed in class is the best choice according to this design objective? Prove that, relative to this objective, the rule you have identified really is better than any of the other voting rules we have seen in class.

(b) Show that the average *expected* regret is not a useful design objective by proving that, as n goes to infinity, all voting rules have the same average expected regret.

Question 4 (10 marks)

Implement the *Kemeny voting rule*, using a programming language of your choice, and test the performance of your implementation for profiles of varying size that have been generated using the *impartial culture assumption*, under which every logically possible preference order is equally likely to occur as the preference order of a given voter. Note that the Kemeny rule is computationally intractable, so a naïve implementation is unlikely to perform well for larger problem instances. A good solution should incorporate at least one modest algorithmic idea that goes beyond a straightforward "brute force" approach based on an exhaustive search. The experimental analysis of your solution should provide some insight into how average runtime depends on, first, the number of voters and, second, the number of alternatives. You may collaborate in groups of up to three people.

For an exercise such as this, I expect solutions to consist of (a) the commented code of your program; (b) clear instructions for how to run your program; and (c) a concise report that explains how you have designed your algorithm and that documents your experimental findings. We should be able to understand your solution (and thus grade your work) by looking only at the report. We will only consult or run your code in case we want to check certain details or verify some of your claims.