15-780: Graduate AI
Homework Assignment #4B Solutions

Out: April 19, 2015
Due: April 22, 2015 5 PM

Collaboration Policy: You may discuss the problems with others, but you must write all code and your writeup independently.

Turning In: Please email your assignment by the due date to shayand@cs.cmu.edu and vdperera@cs.cmu.edu. Your solutions should be submitted as a **single** pdf file. If your solutions are handwritten, **scan** them and make sure they are legible and clear. Please submit your code in separate files and provide instructions on how to run it.

4 Voting

Give an algorithm for manipulating the Borda algorithm and compute its computational complexity. Here we will define the manipulation problem as follows:

> Given a set of candidates C, a distinguished member $c \in C$; and a set V of transitive preference orders on C (e.g. V voters have already given their preferences).
>
> Can an additional voter, given all the above information, choose a preference order P that will ensure that c will be the winner?

The computational complexity should be specified as a function of the above terms (such as $O(|V|)$ or $O(|C||V|)$, etc.) not as simply polynomial or NP, etc. You are free to use research articles on the web, but if so, you must cite your sources.

Compute the total score for each candidate $c' \in C$. Sort the $c' \neq c \in C$ candidates from smallest to largest score. Your preference order places c as the first candidate, and proceeds with the sorted list of candidates. Now add the scores from your preference order to the precomputed scores; if the score for c is the largest, then you can ensure c will be the winner; otherwise, you cannot manipulate the election such that c would be the winner.

The computational complexity is $O(|C||V| + |C| \log |C|)$. This is because computing the scores requires us to add up the contribution from each voter to each candidate's count. This

can be done simultaneously for all the candidates by going through the $|V|$ preference orders once, and since each preference order has $|C|$ elements, this takes $O(|V||C|)$ time. Then we need to sort the candidates according to their scores, which can be done in $O(|C| \log |C|)$ time. Finally, adding the scores from our preference order and checking who has the highest score can be done in $O(|C|)$ time.