Voting Rules in Python

Generating election examples

M2 BDMA
Decision Modelling
Fall 2023
Jose Antonio Lorencio Abril

Election example: a candidate wins all
1st Approach

Theorem

If there are only two profiles and there is a candidate with more than 50% of the votes, then this candidate wins under all voting rules, except maybe Borda.

If the candidate with more than 50% of the votes is in second place in the other profile, then this candidate wins under Borda too.

Proof:

- Plurality: The candidate with more than 50% of the votes wins.
- Plurality with runoff: The candidate with more than 50% of the votes wins.
- Condorcet: If there is a candidate with more than 50% of the votes, it is the Condorcet winner.

Election example: a candidate wins all
1st Approach

Proof:

- Borda:
  - n voters, the **top candidate** has **k votes**, with $k>n/2$
  - The **second top candidate** has then **n-k votes**
  - Top candidate earns **$P1 = n*k+(n-1)*(n-k)$** points
  - Second top candidate earns **$P2 = (n-1)*k+n*(n-k)$**
  - It's easy to reduce **$P1>P2$ to $k>n-k$, which is true because $k>n/2$**.

Using this theorem to generate an example

1. Generate the profile $P1=a>b>c>...$ until having m candidates in the profile
2. Generate the profile $P2=b>a>c>...$ changing the order of the first two candidates
3. Assign $n/2+1$ votes to P1
4. If n is even, assign $n/2-1$ votes to P2; if n is odd, assign $n/2$ votes to P2
5. This way, all conditions are satisfied

Election example: a candidate wins all
1st Approach

Result:

- A>B>C>D>E>F for 21 voters
- B>A>C>D>E>F for 19 voters

Winner is A.

Election example: a candidate wins all
2nd Approach

Random generation

The theorem approach can be boring. There are more sophisticated approaches.

For instance, we can generate elections randomly until all conditions are met.

1. Generate a **random profile with m candidates**, ordered randomly
2. For each voter from 1 to n:
   a. With **probability p, I generate** a new random profile
   b. With **probability 1-p, I add another vote** to the previous profile
3. **Check** the conditions. If they are not met, **repeat**

Election example: a candidate wins all
2nd Approach

Result:

- D>C>B>E>F>A for 4 voters
- B>D>C>F>E>A for 22 voters
- C>B>E>A>F>D for 14 voters

Winner is B.

Election example: a candidate wins all
3rd Approach

Genetic Algorithm

I thought that the random approach might be too inefficient, so I tried to develop a more efficient approach through a GA.

1. Generate the **initial population** of K elections randomly
2. Evaluate the fitness for each election:

   fitness = 3*full_win + 2*req_1 + req_2,

   where

   full_win = 1 if a candidate wins all, 0 otherwise
   req_1 = 1 if no more than 90% of voters have the same preference, 0 otherwise
   req_2 = 1 if no more than 70% of voters have the same best candidate, 0 otherwise
3. **Repeat until there is an election with fitness == 6**:
   a. **Select** best elections
   b. **Crossover** by roulette wheel selection
   c. **Mutation**
   d. Evaluate fitness

Election example: a candidate wins all
3rd Approach

Genetic Algorithm

Selection

By roulette wheel: assign higher probability to those with higher fitness

Crossover

To combine two elections, we merge the two elections in E:

- For each profile in E:
  - new_election[profile] += 1
  - E[profile] -= 1
  - if voters(new_election) = n, break

Example:

Parent1 = {abc:2,bac:1}, Parent2 = {cab:2, bac:1}

E = {abc:2, bac:2, cab:2}

new_election = {abc:1, bac:1, cab:1}

Election example: a candidate wins all
3rd Approach

Genetic Algorithm

Mutation

- If the election has only one profile, divide it into two
- If the election has only two profiles, divide it into three
- Else:
    - Remove the least common profile
    - Add its votes to the most common profile

Election example: a candidate wins all
3rd Approach

Result:

- C>D>F>B>E>A for 18 voters
- E>A>F>B>C>D for 22 voters

Winner is E.

Election example: 4 winners

In this case, I have done:

- The **random approach**: almost the same, change the conditions to check
- The **GA**: almost the same, change the fitness function:

  fitness = 2*n_winners + req_1 + req_2,

  n_winners is the amount of different winners

  In this case, we finish when the fitness is 8.

Election example: 4 winners

Result:

- A>E>F>C>B>D for 3 voters
- C>B>F>A>D>E for 12 voters
- D>A>B>E>F>C for 18 voters
- B>C>F>E>A>D for 7 voters

Winners are:

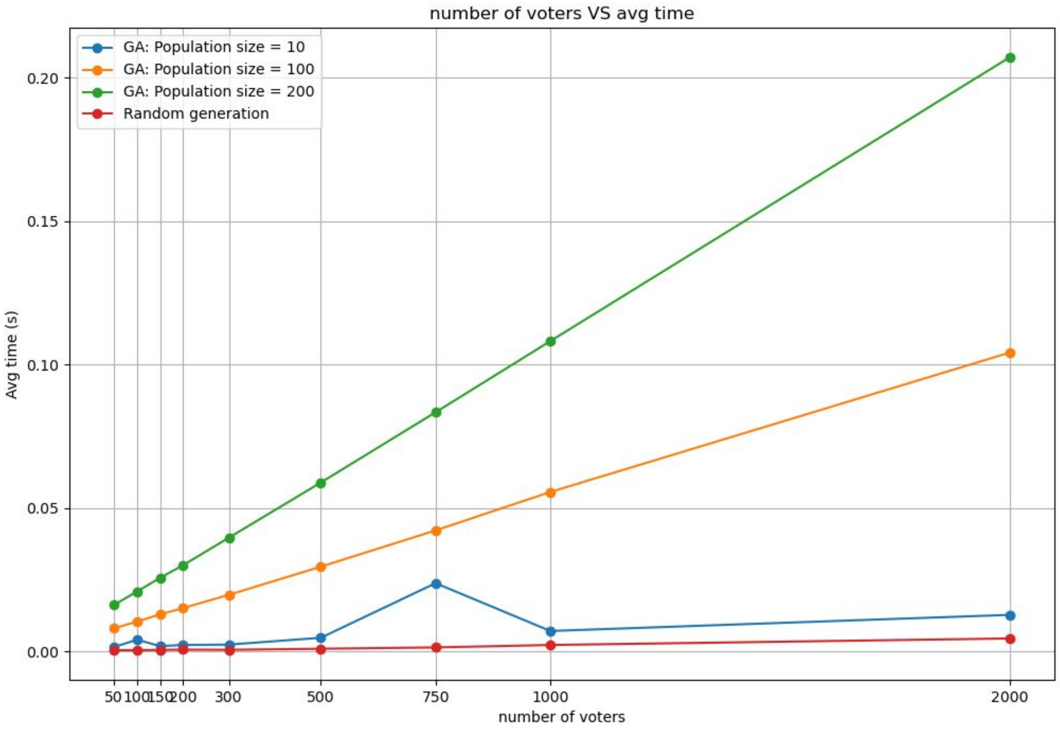
Plurality: D

Plurality Runoff: C

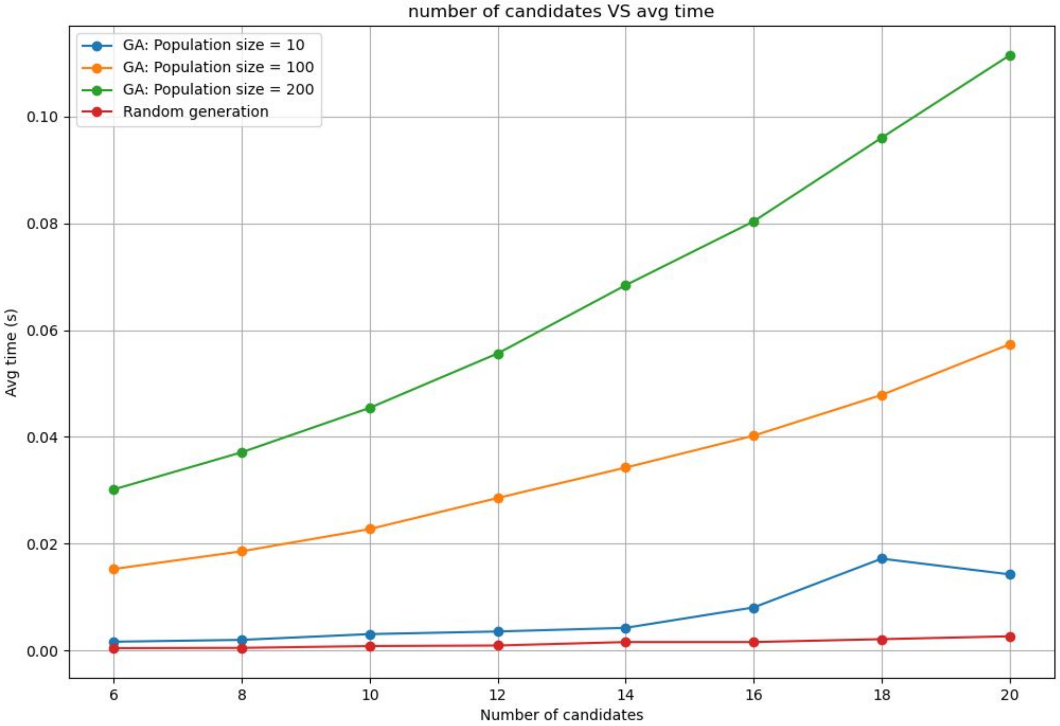
Condorcet: A

Borda: B

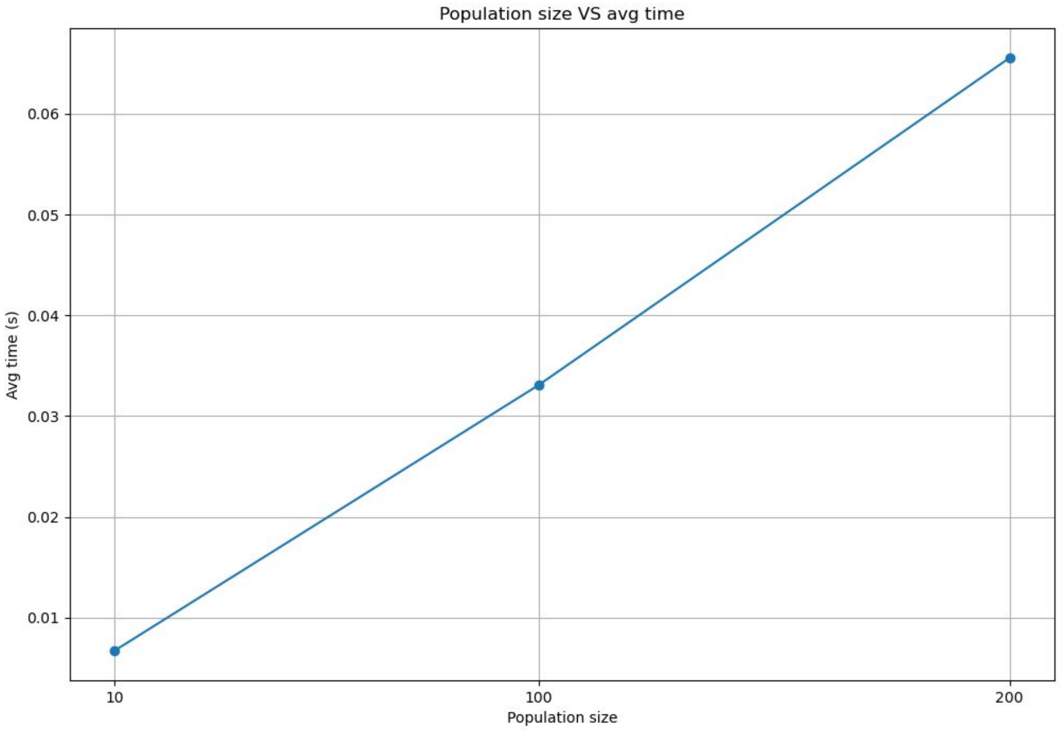
Some performance analysis



Some performance analysis



Some performance analysis



Some performance analysis

The results came out worse than I expected, because I believe that the totally random approach is quite likely to find a solution.

Anyways, it has been interesting to develop the GA method and maybe it can be further improved.